Veröffentlichungen der DGK

Ausschuss Geodäsie der Bayerischen Akademie der Wissenschaften

Reihe C Dissertationen Heft Nr. 898

Stefania Zourlidou

Traffic Regulation Recognition from GPS Data

München 2023

Bayerische Akademie der Wissenschaften

ISSN 0065-5325 ISBN 978-3-7696-5310-6

Diese Arbeit ist gleichzeitig veröffentlicht in:
Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik der Leibniz Universität Hannover
ISSN 0174-1454, Nr. 384, Hannover 2023

Veröffentlichungen der DGK
Ausschuss Geodäsie der Bayerischen Akademie der Wissenschaften

Reihe C Dissertationen Heft Nr. 898

Traffic Regulation Recognition from GPS Data

Von der Fakultät für Bauingenieurwesen und Geodäsie
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des Grades
Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation

von

Stefania Zourlidou, M. Sc.

Geboren am 09.03.1981 in Panorama Thessaloniki, Griechenland

München 2023

Bayerische Akademie der Wissenschaften

ISSN 0065-5325 ISBN 978-3-7696-5310-6

Diese Arbeit ist gleichzeitig veröffentlicht in:
Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik der Leibniz Universität Hannover
ISSN 0174-1454, Nr. 384, Hannover 2023

Adresse der DGK:

Ausschuss Geodäsie der Bayerischen Akademie der Wissenschaften (DGK)
Alfons-Goppel-Straße 11 ● D – 80 539 München
Telefon +49 – 331 – 288 1685 ● E-Mail post@dgk.badw.de
http://www.dgk.badw.de

Prüfungskommission:

Vorsitzender:	Prof. Dr.-Ing. habil. Jürgen Müller, LUH
Referentin:	Prof. Dr.-Ing. habil. Monika Sester, LUH
Korreferenten:	Prof. Dr.-Ing. habil. Christian Heipke, LUH Prof. Dr. Jörg Müller, TU Clausthal-Zellerfeld

Tag der mündlichen Prüfung: 20.03.2023

© 2023 Bayerische Akademie der Wissenschaften, München

Alle Rechte vorbehalten. Ohne Genehmigung der Herausgeber ist es auch nicht gestattet,
die Veröffentlichung oder Teile daraus auf photomechanischem Wege (Photokopie, Mikrokopie) zu vervielfältigen

ISSN 0065-5325
ISBN 978-3-7696-5310-6

Abstract

This dissertation shows the importance of using low-cost crowd-sourced information for the task of traffic regulator recognition (traffic signals, stop signs, priority signs, uncontrolled intersections), the cost of which in terms of time and money is much higher if standard technology is used for surveying. GPS trajectories can reveal the movement patterns of traffic participants, and the initial hypothesis that traffic regulations can be retrieved by mining the movement patterns imposed by traffic rules is verified. The predictive ability of the classifier becomes more accurate when static information derived from open maps (OSM) is merged with dynamic features extracted from GPS trajectories. An extensive evaluation of the proposed methodology on three datasets, provided classification accuracy between 95% and 97%.

For recovering incorrect predictions, an additional consistency check of the predicted regulation labels based on domain knowledge rules is proposed, which increases the classification accuracy by 1%-3%. The low sampling rate of GPS traces was found that can negatively affect the classification performance, decreasing the classification accuracy between 1%-2% when the sampling interval is doubled from 2 s to 4 s. In contrast, excluding curved trajectories from the analysis has a positive effect on classification performance. It was also shown that the optimal number of trajectories per intersection arm in terms of computing classification features is five straight trajectories.

The problem of sparsity of labeled data was investigated by exploring different scenarios of availability of labeled data. Both unsupervised and semi-supervised techniques such as clustering, self-training, cluster-then-label, as well as active learning were examined. The idea of transferability of learning between cities (training a classifier in a city A and predicting regulators in a city B) was also tested, discovering the conditions under which it may be feasible and its limitations. The most accurate predictions of the above tested learning methods were achieved through active learning, which was found to reduce the number of labeled data required for training by 66.7% in the two tested datasets.

Finally, a hypothetical scenario is described for the first time, to the author's knowledge, in the field of traffic regulation recognition from GPS data, where information arrives as data streams, opening up further the possibilities to address the traffic regulation recognition problem in an incremental and more dynamic way.

Keywords: traffic-regulations, crowd-sourcing, GPS tracks

Kurzfassung

Diese Dissertation zeigt die Wichtigkeit der Nutzung von kostengünstigen Crowd-Sourced-Informationen für die Erkennung von Verkehrsregulatoren (Ampeln, Stoppschilder, Vorfahrtsschilder, ungeregelte Kreuzungen) auf, deren Kosten in Form von Zeit und Geld viel höher sind, wenn Standardtechnologie für die Vermessung verwendet wird. GPS-Trajektorien können die Bewegungsmuster von Verkehrsteilnehmenden aufzeigen, und die anfängliche Hypothese wurde bestätigt, dass Verkehrsregulatoren durch die Auswertung von Bewegungsmustern, die durch Verkehrsregeln vorgegeben sind, ermittelt werden können. Die Vorhersagefähigkeit des Klassifikators wird genauer, wenn statische Informationen aus offenen Karten (OSM) mit dynamischen Merkmalen aus GPS-Trajektorien kombiniert werden. Eine umfassende Bewertung der vorgeschlagenen Methodiken am Beispiel von drei Datensätzen ergab eine Klassifizierungsgenauigkeit zwischen 95% und 97%.

Um falsch klassifizierte Regulatoren zu korrigieren, wird eine zusätzliche Konsistenzprüfung der vorhergesagten Bezeichnungen auf der Grundlage von Regeln auf Basis des Domänenwissens vorgeschlagen, die die Klassifizierungsgenauigkeit um 1%-3% erhöht. Darüber hinaus wurde festgestellt, dass die niedrige Samplingrate von GPS-Tracks die Klassifizierungsleistung negativ beeinflussen kann und die Klassifizierungsgenauigkeit um 1%-2% sinkt, wenn das Samplingintervall von 2 auf 4 Sekunden verdoppelt wird. Außerdem wurde festgestellt, dass der Ausschluss von gekurvten Trajektorien aus der Analyse einen positiven Effekt auf die Klassifizierungsleistung hat. Zusätzlich wurde festgestellt, dass die optimale Anzahl von Trajektorien pro Kreuzungsarm im Hinblick auf die Berechnung von Klassifizierungsmerkmalen fünf gerade Trajektorien sind.

Die Problematik des Fehlens gelabelten Daten wurde untersucht, indem verschiedene Szenarien der Verfügbarkeit von gelabelten Daten erforscht wurden. Hierbei wurden sowohl unsupervised learning als auch semi-superivised Techniken wie clustering, self-training, cluster-then-label sowie active learning untersucht. Die Idee der Übertragbarkeit des Lernens zwischen Städten (Training eines Klassifizierers in einer Stadt A und Vorhersage von Regulatoren in einer Stadt B) wurde ebenfalls getestet. Dabei wurden sowohl die Bedingungen der Machbarkeit, als auch die Grenzen aufgedeckt. Die genauesten Vorhersagen der oben getesteten Lernmethoden wurden durch active learning erreicht, welches die Anzahl der für das Training erforderlichen gelabelten Daten in den beiden getesteten Datensätzen auf 66.7% reduziert.

Abschließend wird ein hypothetisches Szenario beschrieben, das nach Kenntnis der Autor zum ersten Mal im Bereich der Erkennung von Verkehrsregulatoren anhand von GPS-Daten auftritt. Hierbei treffen die Informationen in Form von Datenstreams ein, was weitere Möglichkeiten eröffnet, das Problem der Erkennung von Verkehrsregelungen auf inkrementelle und dynamischere Weise anzugehen.

Schlagworte: Verkehrsregulatoren, Crowd-Sourcing, GPS-tracks

Contents

1. Introduction **1**
1.1. From GPS tracks to Traffic Regulations 1
1.2. Motivation for Learning Intersection Traffic Regulations 4
1.3. Research Gap 6
1.4. Motivation for Learning Traffic Regulations from GPS Data 6
1.5. Research Objectives, Challenges and Contributions 12
1.6. Outline of the Thesis 14
1.7. Summary 15
1.8. Acknowledgements 15

2. Theoretical Background **17**
2.1. Intersections and Intersection Traffic Regulations 17
  2.1.1. Intersections 17
  2.1.2. Intersection Traffic Regulations 20
2.2. Spatiotemporal Data and Movement Trajectories 22
  2.2.1. The Global Positioning System 22
  2.2.2. Sampling Frequency 24
  2.2.3. The GPS Exchange Format: GPX 25
  2.2.4. Movement Patterns in Spatiotemporal Data 27
  2.2.5. Spatiotemporal Data Mining 29
  2.2.6. Semantic Enrichment of Trajectories 32
  2.2.7. Detecting Stop and Moves: the CB-SMoT Algorithm 34
2.3. Machine Learning 40
  2.3.1. Machine Learning and Types of Learning 40
  2.3.2. Supervised-Learning: Classification 42
  2.3.3. Unsupervised-Learning: Clustering 52
  2.3.4. Semi-supervised Learning 57
  2.3.5. Active-Learning 59
  2.3.6. Incremental Learning 61
2.4. Acknowledgements 63

3. Related Work **65**
3.1. Existing Traffic Regulation Recognition Approaches 65
3.2. Static Categorization 68
  3.2.1. Map-based Category 68

3.2.2. Image-based Category 69
3.3. Dynamic Categorization 69
3.3.1. Episode-based Category 69
3.3.2. Speed-profile Category 71
3.3.3. Movement-summarization Category 72
3.4. Hybrid-based Categorization 73
3.5. Discussion 74
3.6. Knowledge Gap 76
3.7. Acknowledgements 77

4. Traffic Regulation Recognition (TRR) from GPS Data 79
4.1. Introduction 79
4.2. Datasets 79
4.2.1. Dataset Requirements and Limitations 79
4.2.2. Datasets for Testing the Proposed Methods 80
4.2.3. Groundtruth Map Construction 82
4.3. Methodology 84
4.3.1. Detection of Stop and Deceleration Episodes 84
4.3.2. The Static Approach 87
4.3.3. The c-Dynamic Approach 88
4.3.4. The Dynamic Approach 92
4.3.5. The Hybrid Approach 93
4.3.6. Implementation and Classification Settings 93
4.4. Results 94
4.5. Discussion 95
4.6. Summary 96
4.7. Acknowledgements 96

5. TRR From GPS Data: One-Arm versus All-Arm Models 97
5.1. Introduction 97
5.2. Methodology 97
5.2.1. One-Arm vs. All-Arm Models 97
5.2.2. The Effect of Sampling Rate 98
5.2.3. Reduced Models 98
5.2.4. The Effect of Turning/No-Turning Trajectories 98
5.2.5. The Effect of Number of Trajectories 99
5.2.6. Application of Domain Knowledge Rules 99
5.2.7. Classification Settings 102
5.3. Results 102
5.3.1. One-arm vs. All-arm Models 102
5.3.2. Testing the Effect of Sampling Rate 105

5.3.3. Reduced Models . . . 105
5.3.4. Testing the Effect of Turning Trajectories and Examining an Optimal Number of Trajectories . . . 106
5.3.5. Testing the Effect of the Number of Trajectories on Classification Performance . . . 109
5.3.6. Misclassification Analysis . . . 111
5.3.7. Applying Domain Knowledge Rules . . . 115
5.4. Discussion . . . 118
5.5. Summary . . . 119
5.6. Acknowledgements . . . 120

6. TRR with Sparsely Labeled and Stream Data 121
6.1. Introduction . . . 121
6.2. TRR with Clustering . . . 121
6.3. TRR with Self-Training, Active Learning and Cluster-then-Label . . . 124
6.3.1. TRR with Self-Training: Using Labeled and Unlabeled Data . . . 124
6.3.2. TRR with Active Learning . . . 125
6.3.3. TRR with the Cluster-then-Label Algorithm . . . 127
6.3.4. Comparison of All Tested Methods . . . 128
6.4. Learning Transferability: Training on City A and Predicting on City B . . . 129
6.5. Incremental (Online) Learning . . . 137
6.6. Summary . . . 141

7. Conclusions and Outlook 143
7.1. Research Questions Addressed in this Thesis . . . 143
7.2. Outlook . . . 145

List of Acronyms 146

Index 147

A. Appendix 151

List of Figures 161

List of Tables 167

Bibliography 169

Curriculum Vitae 179

Acknowledgements 180

1. Introduction

1.1. From GPS tracks to Traffic Regulations

The practice of exchanging (creating and sharing) geographical information has grown exponentially in recent years. The reason for this is the proliferation of the Internet and the World Wide Web, and the development of geographic information technologies. Compared to the early disorganized era of “sharing”, where the primary providers of geographic information were national governments, the modern phase, already characterized as “chaotic” in 2007 by Goodchild et al., in terms of the network of producers and consumers, remains so. Nowadays, individuals can collect and share data or information of various kind from their environment: news (Rodosthenous and Michael, 2021), photos (Depauw et al., 2022), speed (Hu et al., 2016) and noise measurements (Picaut et al., 2019), air pollution (Chen et al., 2018), atmospheric (Muller et al., 2015) and precipitation data (Fitzner and Sester, 2016), etc. These data and information that are associated with a location relative to the earth are called *geodata*, also known as *geographic* data or *geospatial* data. From these, interesting information about a particular phenomenon that occurred at a particular location for a given time or period of time can be estimated, such as the examples just mentioned.

Moreover, all smartphones are now equipped with GPS chips, accelerometers and gyroscopes and their increasing usage has further expanded the possibilities for *Spatial Crowdsourcing (*SC), a term describing “the potential of the crowd to perform real-world tasks with strong spatial nature that are not supported by *Conventional Crowdsourcing (*CC) techniques” (Gummidi et al., 2019). CC techniques lack the spatial element and focus on transactions that are conducted entirely over the Internet. In contrast, SC requires on-site physical presence and this collected information has increasing potential (Heipke, 2010) and significance in practice, for example, it is often used as part of the innovation process of organizations (Karachiwalla and Pinkow, 2021). More specifically, the field of crowdsourcing has lots of potential for mapping mainly due to two reason: (a) the technology is mature enough to stay for long, and (b) other means of mapping are too slow or too expensive (Heipke, 2010). Guo et al. (2014), discussing mobile *crowd-sensing*, distinguish two unique features: (1) it involves both implicit and explicit participation and (2) it collects data from two user-participant data sources - mobile social networks and mobile sensing. Either with minimum (opportunistic) or significant (participatory) awareness and involvement, in a crowd-sensing system volunteers can quickly gather a lot of geographic information that can then be processed for many purposes (Lane et al., 2008).

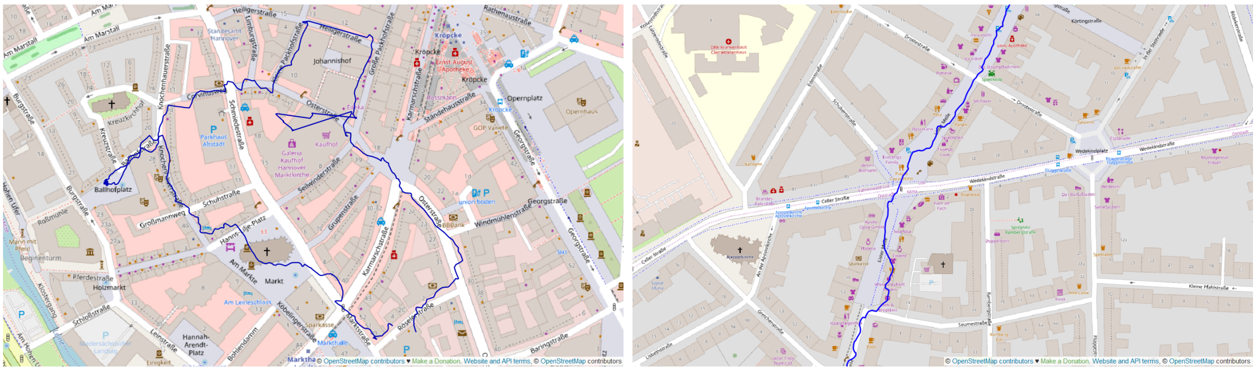
There are numerous examples of how crowdsourced information from individuals can be put to practical use. Social media crowdsourcing can increase our understanding of human dynamics and spatio-temporal characteristics of cities and convey information about cities (Zhou and Zhang, 2016). For example, city locations (hotspots) that attract different human activities at various intensities at different time intervals. Other popular topics based on such data include extracting context information in the form of *interesting places* (Agamennoni et al., 2009), computing speed estimates for accurate trip arrival prediction (Niehöfer et al., 2009) and mining road features such as road names and classes (Li et al., 2015). Furthermore, using common devices such as smartphones with low cost sensors, predictions of phenomena such as earthquakes (early warning), which previously required special equipment, can now be implemented (Minson et al., 2015).

Urban sensing has become increasingly popular for applications ranging from real-time detection of traffic lights (Zhu et al., 2013) to the detection of road/traffic conditions (bumps, hard braking, honking, roughness of bicycle paths) based on either mobile devices (Xiao et al., 2021; Mohan et al., 2008; Wage and Sester, 2021) or cars equipped with sensors (Eriksson et al., 2008; Fox et al., 2017). Drivers in large cities can be informed about the existence of vacant parking spaces near their destination, a citizen service realizable using crowdsourced data (Salpietro et al., 2015; Mathur et al., 2010). Finally, by analysing the aggregated acoustic signal collected from the user's smartphone microphone sensor, the traffic state of the streets can be estimated (Vij and Aggarwal, 2018).

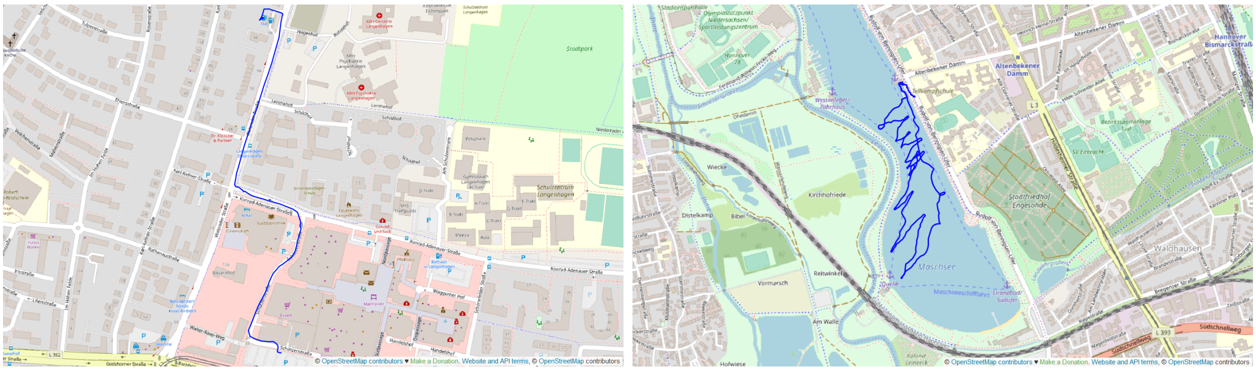
Some of the aforementioned approaches use data from various sensors to achieve their goals. However, the minimum data required to extract spatial information remains the position of an object or event (GPS log). A grade slightly above this minimum data is the temporally ordered positions of a moving object that when connected make up a *spatiotemporal trajectory*. These positions are obtained as sequences of GPS records. Some examples of trajectories of various moving objects (pedestrian, bicycle, car, sailboat) are shown in Figure 1.1. The movement of objects carrying GPS recording devices can be recorded and the interpretation of their recorded trajectories can be revealing either about the location/time/time interval in which the motion takes place or about the moving objects themselves. This is because the trajectories contain implicit knowledge of object movement regarding the underlying patterns and structure of the movement that can be identified by pattern recognition techniques and then applied to various domains (Sester et al., 2012). For example, human activities can be predicted from pedestrian or vehicle trajectories (Makris and Ellis, 2002, 2003; Hu et al., 2004; Piciarelli and Foresti, 2005) and transportation routes can be optimised by processing trajectory data from urban transport (Feng and Zhu, 2016).

Focusing on vehicle-derived trajectories, some examples of their application are the inference of the condition of traffic and the recognition of traffic patterns (Guo, 2008; Atev et al., 2006), as well as the estimation of intersection travel time (Tang et al., 2016). Regarding movement patterns, a conceptual view of trajectories (semantic trajectories) by detecting stops and movements (Spaccapietra et al., 2008; Bermingham and Lee, 2018) allows us to discover hidden patterns in the movement of objects (Alvares et al., 2007, 2009; Palma et al., 2008) such as pattern relationships or to reveal interesting locations for individuals or groups of people (Spinsanti et al., 2010; Phithakkitnukoon et al., 2010). Other trajectory-related applications involve inferring trip purposes (Gong et al., 2016), detecting sudden events such as braking incidents (Dang et al., 2014), classifying road user behaviour (Laureshyn et al., 2009) and detecting anomalies (outliers) in traffic (Jiang et al., 2009) or in route navigation (Huang et al., 2014).

One area of great research interest recently is automatic map updating, especially in view of recent developments in autonomous driving. Moreover, the increasing use of navigation devices nowadays that help drivers to quickly reach their destination is another factor that makes up-to-date and accurate maps important. For this reason, many research studies focus on how maps can be updated given the frequent changes that take place in the road network. According to Mapscape (2022), roads change up to 15 percent annually. Changes to the road network can be either temporary or permanent, i.e., temporary closures due to construction projects or natural disasters, and permanent such as new roads and changes



(a) Walking *(b)* Cycling



(c) Car driving *(d)* Sailing

Figure 1.1.: Trajectories from moving objects in different movement modes: (a) walking, (b) cycling, (c) car driving and (d) sailing.

to sections of existing roads, to name a few. The map update refers to both the road network itself and the various features that come on top of it. The research interest in this area lies basically in extracting the geometric and topological features of the road network. Numerous studies have focused on the automatic generation of the road network from GPS traces (Davies et al., 2006; Cao and Krumm, 2009; Biagioni and Eriksson, 2012; Ahmed et al., 2015; Mariescu-Istodor and Fränti, 2018), on the detection of changes in the road network (Tang et al., 2019; He et al., 2018) and in related subtopics such as the detection of intersections (Fathi and Krumm, 2010; Wu et al., 2013; Wang et al., 2017). The innovative element behind these map inference methods is that they are not based on data obtained with special survey equipment, a process that is time-consuming and expensive, but on crowdsourced GPS traces, recorded by vehicles with simple GPS devices. Thus, using only spatio-temporal samples (longitude-latitude-time) from vehicles that act as probes, both the geometry and connectivity of road segments can be obtained rapidly.

However, this interest on automatic road network extraction from GPS tracks, expressed by the vast literature, is not uniform for all categories of map features that also need to be automatically updated using GPS tracks. One such relatively unexplored research area is that of *traffic regulations* (e.g., traffic signs). In the next section we explain why traffic regulations are important map elements and what practical applications can benefit from having them on maps.

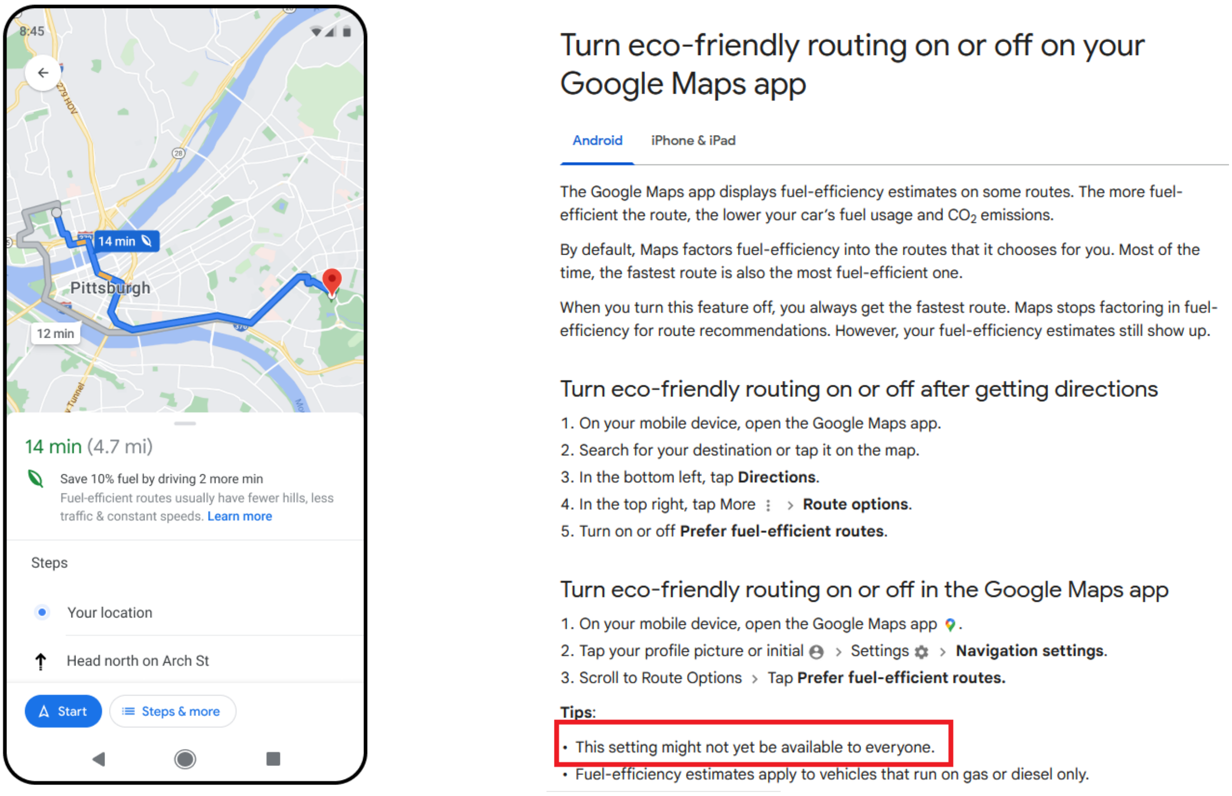
1.2. Motivation for Learning Intersection Traffic Regulations

Road network traffic regulations, also called *traffic controls*, *rules* or *regulators*, are used to control the traffic of road users such as vehicles, bicycles and pedestrians at intersections. Traffic regulations at intersections contain important navigational information and could, for example, help in deriving more accurate travel time estimates. Traffic regulators, such as traffic lights, have a significant impact on traffic flow at intersections, which in turn contribute to increased fuel consumption because they involve queuing conditions and many stop-and-go incidents. According to Alshayeb et al. (2021), intersections are one of the main places where an excessive amount of fuel is consumed. In addition, traffic lights contribute more to air pollution compared to other types of controls due to exaggerate vehicle emissions at these locations (Gastaldi et al., 2014). Therefore, for fuel-efficient and *green* map applications (Ganti et al., 2010; Saremi, 2016; Zhao et al., 2017), it is important to have this traffic regulator related information on the maps and be up to date. Google maps recently released (November 2021) the option of *eco-friendly routing* (Figure 1.2), which until the writing of this dissertation is not available to everyone[1]. According to Google Maps (2022), the eco-routing after generating a set of potential route candidates, incorporates data about road conditions, such as road closures, live traffic-data and historic traffic patterns, into machine learning algorithms to accurately predict the traffic during the course of a user's journey. In this way, under user's demand, route recommendations also take into account expected fuel consumption. As previously discussed, intersection traffic regulations can also be included in the list of data with which such predictions are made. In addition, the type of regulators controlling intersections is also crucial for the development of autonomous vehicles and more specifically for the decision making involved on how they drive through intersections and interact with other road users (pedestrians, cyclists, other autonomous or human-driven vehicles).

Moreover, car safety is a matter of great interest to both drivers and car manufacturers. Advance Driver Assistance Systems (ADAS) aim to assist drivers in the complex driving process by detecting pedestrians, facilitating parking, helping with lane changes and crossing intersections, recognising traffic signals and adjusting speed to avoid collisions, to name a few. One idea for improving the performance (e.g., by reducing the false positive alerts) of such safety-oriented systems is to fuse data from maps and other vehicle sensor sources. For example, Peker et al. (2014) propose fusing map features (type of driving maneuvers allowed, types of intersections, road network characteristics, etc.) with images obtained from in-vehicle cameras to detect traffic signs with higher probability. Satzoda et al. (2013) show how a multimodal synergistic approach can be applied for automated driving analysis by analyzing data from different sources (CAN Bus, map and GPS devices) in a cooperative and complementary way. Lefèvre et al. (2011, 2012) predict driver intention and assess risk at road intersections based not only on vehicle kinematics and dynamics, but also on contextual information in the form of topological and geometrical characteristics of the intersections (traffic lanes, intersection connections), as well as hidden behavioral variables (the set of authorized maneuvers and traffic rules (stop, yielding, etc.)), which are derived from digital maps. The performance of the proposed methods in different test scenarios shows an improvement over the results of similar methods based on single-source input data only (without using traffic rules, lanes, etc.).

---

[1] The author last checked the Google maps settings of an Android device on 09.08.2022.



14 min (4.7 mi)

Save 10% fuel by driving 2 more min

Fuel-efficient routes usually have fewer hills, less traffic & constant speeds. Learn more

Turn eco-friendly routing on or off on your Google Maps app

Android iPhone & iPad

The Google Maps app displays fuel-efficiency estimates on some routes. The more fuel-efficient the route, the lower your car's fuel usage and CO_2 emissions.

By default, Maps factors fuel-efficiency into the routes that it chooses for you. Most of the time, the fastest route is also the most fuel-efficient one.

When you turn this feature off, you always get the fastest route. Maps stops factoring in fuel-efficiency for route recommendations. However, your fuel-efficiency estimates still show up.

Turn eco-friendly routing on or off after getting directions

1. On your mobile device, open the Google Maps app.
2. Search for your destination or tap it on the map.
3. In the bottom left, tap **Directions**.
4. In the top right, tap More ⋮ > **Route options**.
5. Turn on or off **Prefer fuel-efficient routes**.

Turn eco-friendly routing on or off in the Google Maps app

1. On your mobile device, open the Google Maps app.
2. Tap your profile picture or initial > Settings > **Navigation settings**.
3. Scroll to Route Options > Tap **Prefer fuel-efficient routes.**

Tips:

- This setting might not yet be available to everyone.
- Fuel-efficiency estimates apply to vehicles that run on gas or diesel only.

(a) (b)

Figure 1.2.: (a) Google maps eco-friendly routing. (b) Eco-routing is still not available to everyone[2].

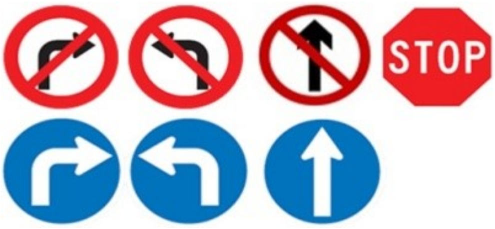


Figure 1.3.: Some examples of traffic signs.

In addition, in the context of navigation, safety issues are in most cases limited to speed limit reminders and speed limit alerts. For this reason, maps that include the local regulatory framework could help to enhance safety (Zourlidou and Sester, 2015a,b). Focusing on the intersection rule context, *regulation-aware* maps, i.e., maps containing traffic regulators, could help drivers to cross intersections safely. Some examples of such traffic regulations, indicated by traffic signs, are illustrated in Figure 1.3.

Finally, another need for mapping intersection traffic regulations comes from the progression of Location Based Services (LBS), 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. Krisp and Keler (2015) go one step further of the existing personalized navigation and way-finding services and propose the idea of

---

[2] Image Sources: `https://www.gstatic.com/gumdrop/sustainability/google-maps-eco-friendly-routing.pdf` and `https://support.google.com/maps/answer/11470237?hl=en&ref_topic=3292869` accessed 09.08.2022.

providing routing suggestions to drivers that avoid *complicated crossings* in urban areas. For example, a complicated intersection may be an intersection where the road network is intersected by bike lanes and tram rails. Similarly, a complicated crossing can be a left turn at an intersection that is not controlled by a traffic light. Conversely, an easy intersection may be an all-way stop controlled intersection. Such recommendations can be useful to inexperienced drivers or drivers who for whatever reasons do not feel comfortable driving in an urban environment. Traffic regulations could also be explored in the context of road driving complexity and incorporated into the definition of rating intersections' complexity for personalized route recommendations.

In summary, we identified four possible applications that can be realised if intersection traffic regulations are featured on maps: 1) fuel-efficient routing, 2) autonomous vehicles, 3) ADAS and 4) personalized routing. In the next section we reveal the research gap in the areas of automatic map update and traffic regulation recognition.

1.3. Research Gap

When examining National Agencies maps or public map databases, such as OpenStreetMap (OSM), one can notice that most intersections are missing information on traffic regulations. And even if this information is included, it concerns mainly traffic signals (traffic lights). Figure 1.4 shows the OSM from a very central and busy area of a large city in northern Germany, Hanover, where traffic regulation information is available for only five of the fifty intersections shown, all five controlled by traffic lights. This is remarkable, as in this city there is a very active OSM community, which provides very detailed OSM maps in general. Moreover, as Hu et al. (2015) points out, despite the importance of traffic regulations, there is currently no national database of the traffic regulations of intersections, unlike the case of road maps. Instead, this information is fragmented in physical records of various municipal authorities.

Consequently, having explained in the previous Section 1.2 the importance of knowing the traffic regulations of intersections and having noted here the insufficient indication of traffic regulators on maps, the need for new methodologies that can collect this information massively, quickly, accurately and at low cost becomes obvious. Since the most accurate way to collect this information is to use surveying equipment, which is neither fast nor cheap, the remaining alternative is to *learn* or *predict* traffic regulations from available data sources. Such data can be crowdsourced images or GPS trajectories. In the next section, we argue about the advantage of using GPS data compared to images.

1.4. Motivation for Learning Traffic Regulations from GPS Data

The task of automatic detection and identification of traffic regulators can be solved by using different data sources. Related studies, mainly use images or GPS traces (Zourlidou and Sester, 2019). Images obtained from cameras or mobile mapping systems can be used to create a traffic sign inventory system. However, acquiring images with such equipment has high time and operational costs. Another option could be to use street-level images offered by platforms such as Google Street View (2022) and Mapillary (2022). The use of Google Street images would have cost constraints, as access to the associated API for large-scale use would require a license. Also, as Hu et al. (2015) point out, there are still many cities and

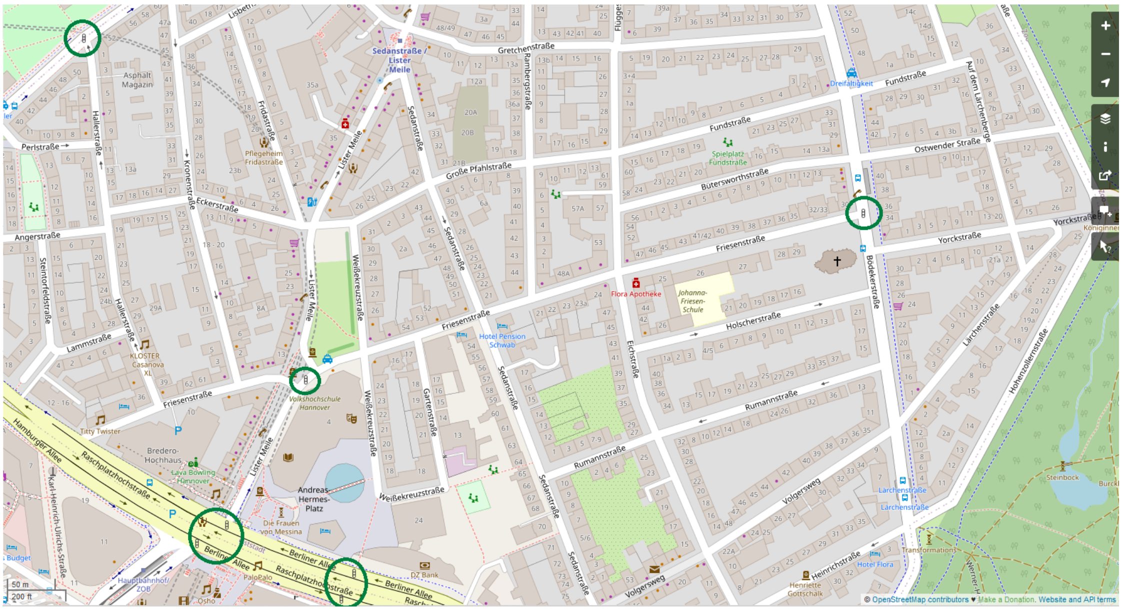


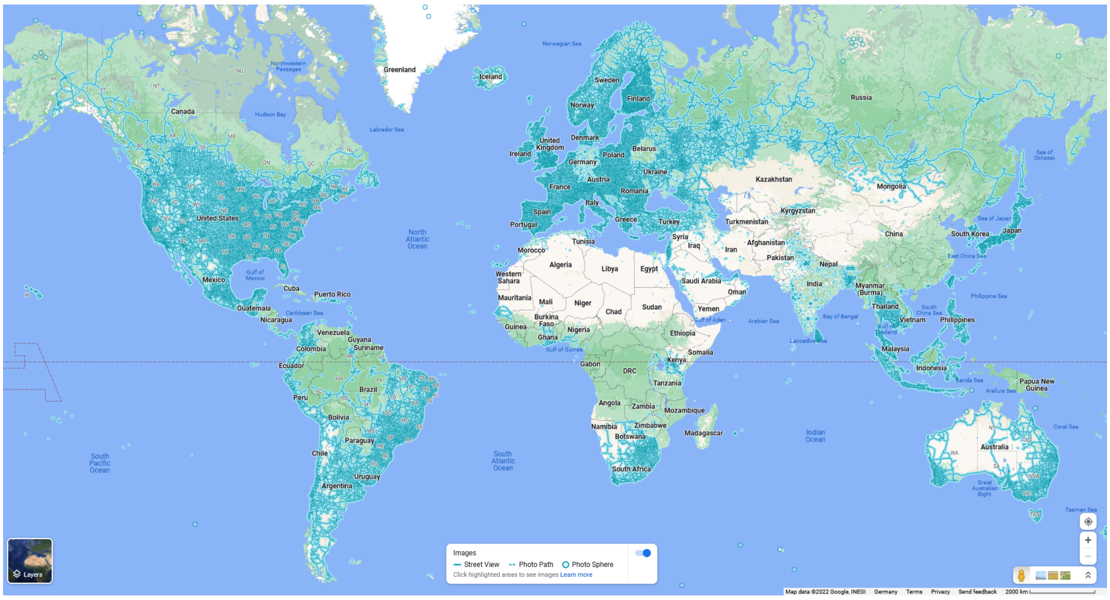
Figure 1.4.: Intersections for which OSM contains traffic regulation relevant information (in green circles). For all other intersections no such information is present. Accessed on 04.08.2022.

places that are not covered by these services and therefore there are no images available to be crawled for traffic regulation detection. Recent quantitative descriptions of road network coverage for both providers were not found. However, we can assess the coverage visually from Figure 1.5 and 1.6, that show the current road network coverage of the two main road image providers mentioned above. In Google Street View (Figure 1.5), 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). Similarly, although Mapillary's data is free of charge[3], its images have similar limitations to those of Google Street View. For example, although many streets in Germany are covered by photos (Figure 1.6a), the coverage mainly concerns main streets. Figure 1.6b shows the central area of Hanover, where one can see that is not fully covered and areas near the center are covered even more sparsely (Figure 1.6c).

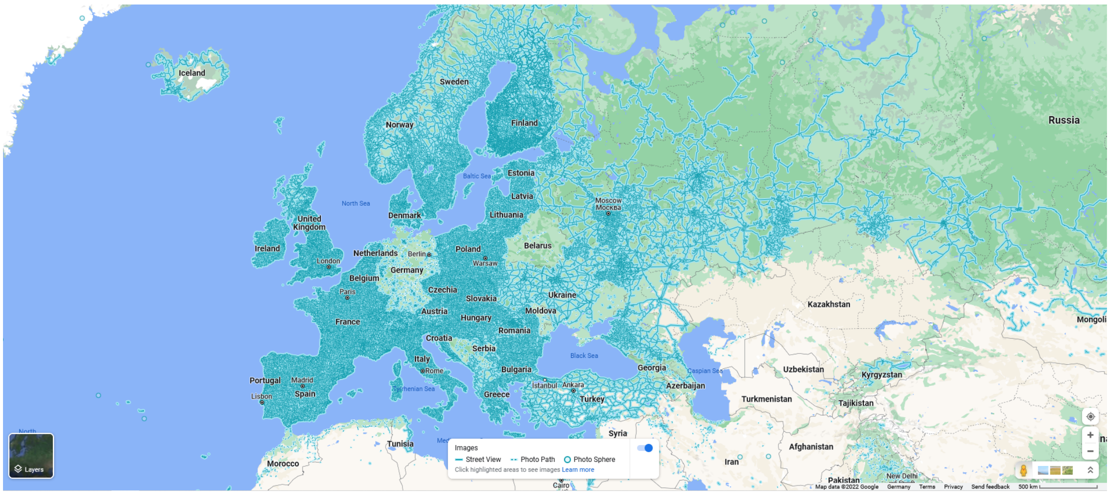
Traffic sign recognition from vehicle cameras is a popular and well-established topic in the computer vision community, providing accurate recognition of traffic signs (Huang et al., 2017; Ardianto et al., 2017). However, although modern cars have cameras, manufacturers do not share their data. But even if such data were available, their adoption in a crowd-sourced scenario would have limitations. One drawback is the generation of large volumes of data (images) and therefore the consumption of resources such as bandwidth and storage space. Also, the cameras must be mounted on vehicles, adding further constraints for broad user participation. According to Kosonen and Henttonen (2015), the key issue of the launching stage of a crowd-sourcing project is to convince people to start volunteering, and for persuading them, it is important to clearly demonstrate how easy it is to participate. In a hypothetical scenario of crowd-sourcing images for traffic regulation recognition with cell phone cameras, people would need to place their cell phones on a car cell phone holder

---

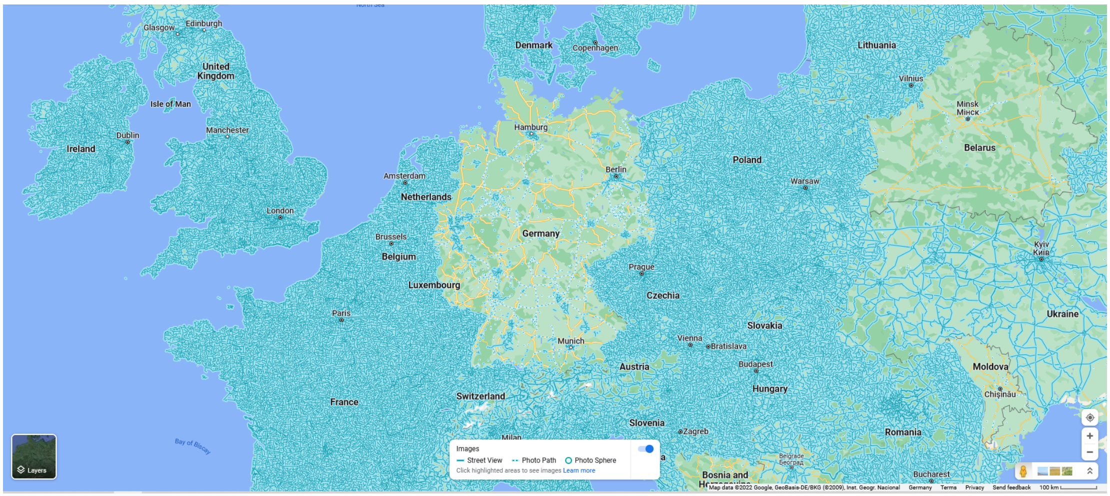
[3] https://help.mapillary.com/hc/en-us/articles/360020754199-Pricing. Accessed 08.08.2022 for checking the pricing costs.



(a) World wide.

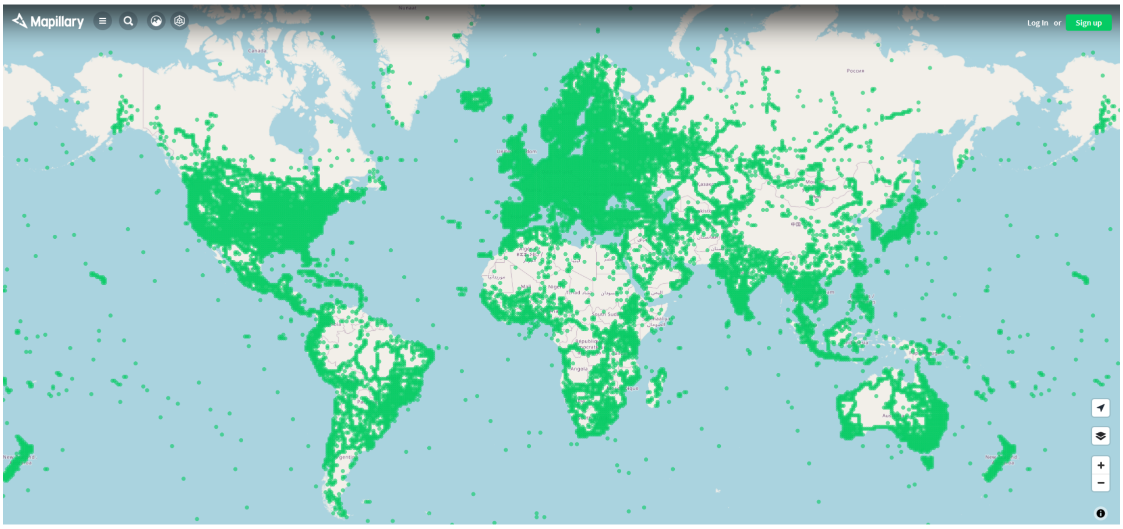


(b) Europe.

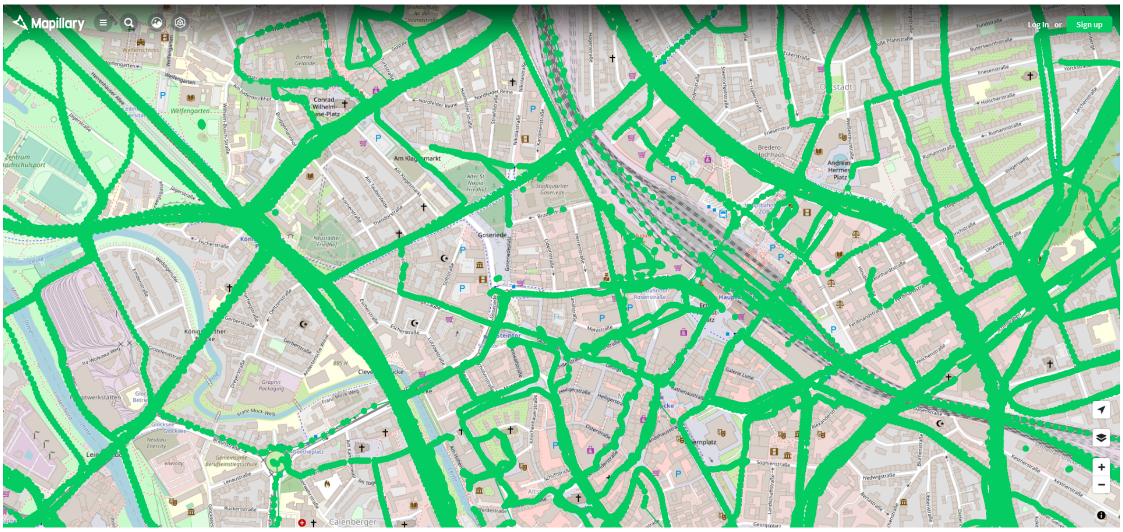


(c) Germany.

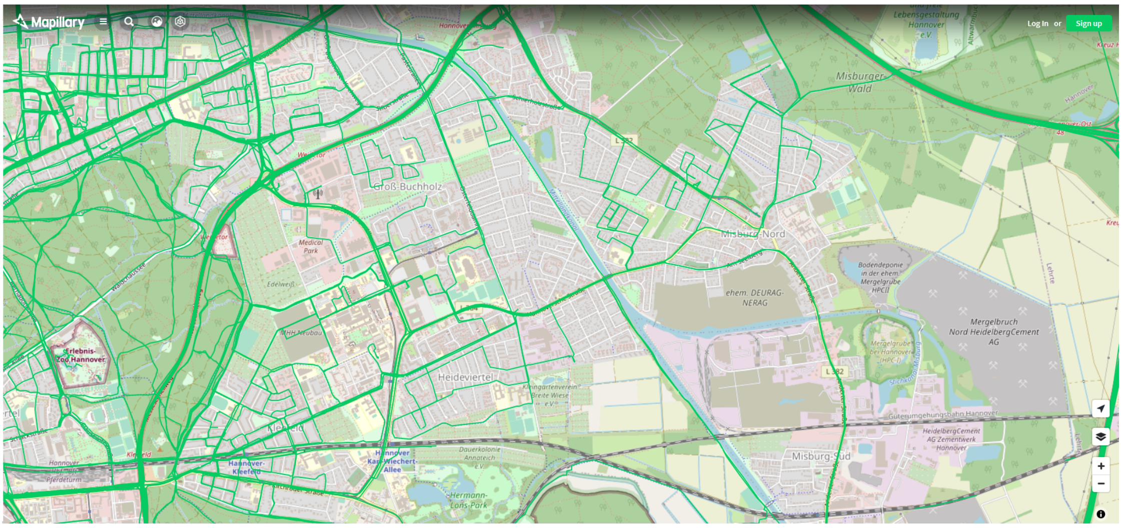
Figure 1.5.: Google Street View coverage (blue lines). The images were accessed on 09.08.2022.



(a) World wide.



(b) Center of Hanover city, Germany.



(c) Non central area in Hanover, Germany.

Figure 1.6.: Coverage of Mapillary Street level Imagery (green lines). The images were accessed on 10.08.2022.

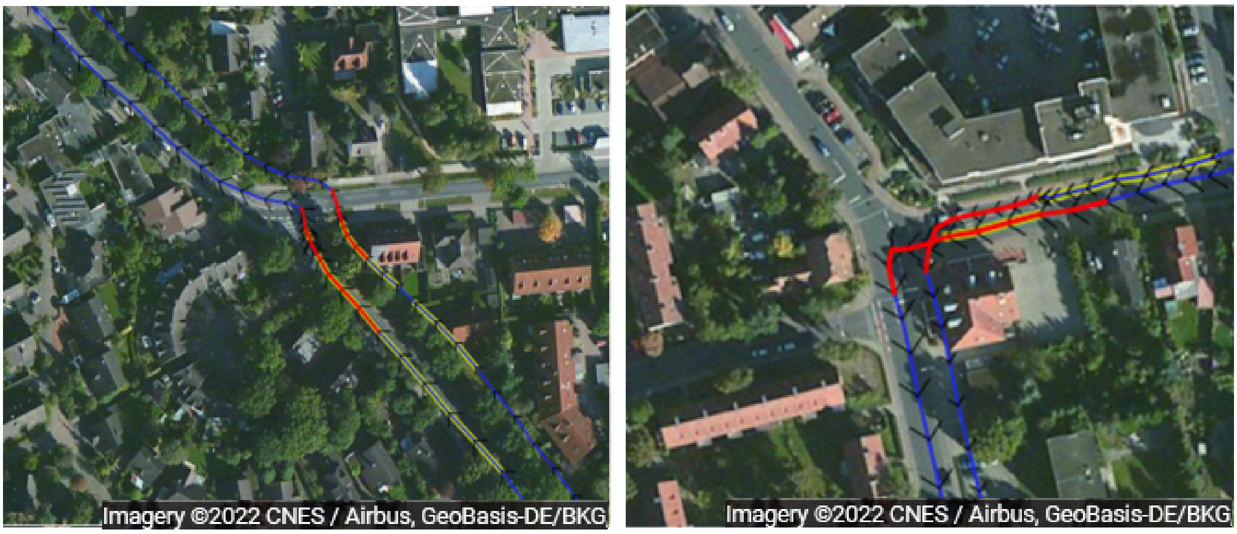
each time they drive, make sure the camera lenses and car window are clean, check that the phone's battery is charged or plugged in a phone charger (battery consumption is very high when taking photos), and address other possible issues such as whether there is enough data storage or internet data available 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 (Balali and Golparvar-Fard, 2016).

The second, more time and cost friendly solution for massive intersection traffic regulator recognition are GPS traces. GPS traces (i.e., time-ordered sequences of recorded locations) are a "lightweight" data source in terms of battery and storage requirements. A file containing the locations and timestamps of a 20-minute car trip, sampling the location every 2 seconds, is only 15 KBytes in size. GPS tracks can also be recorded without special equipment. All smartphones today have GPS receivers and can therefore be used to record movements. According to Merry and Bettinger (2019), an iPhone 6 (released in September 2014) has an average position accuracy of 7-13 meters in an urban environment, for different settings of season, time of day, and WiFi usage period. Since GPS trace recording does not require the phone's display, which is a power-intensive component of the phone, to be on, battery consumption is only slightly affected. In addition, GPS traces require minimal or no user intervention in the recording task (launching a recording application or running in the background when the phone is started). Another advantage over image recording is that they do not impose special requirements on the placement of the device, as is the case with a camera (installation on the front car window on a stable phone mount). Therefore, GPS data is considered a more user-friendly solution for collecting data under a crowd-sourcing scenario for recognizing intersection traffic regulations.

Another free solution for the recognition of traffic regulators is to use open data obtained from OSM (Saremi and Abdelzaher, 2015). Besides the map features that explicitly stand in OSM, such as shops, buildings, roads, etc., there is also implicit information that can be extracted and exploited for the classification task of traffic regulators. Such information, as discussed later in the methodology section of this thesis, can be attributes describing the connectivity of intersections, such as the length of the road to which an intersection belongs and the distances of an intersection from neighboring intersections. This information can be freely extracted from OSM and then fed to a classifier to predict intersection traffic controllers. However, the predictive results have certain limitations, as will be explained later. Therefore, for the objectives of this thesis, we chose to use GPS traces of vehicles combined with open data obtained from OSM.

The idea of using GPS data to identify intersection traffic controls 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 trajectories that cross 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 a stop sign under certain circumstances, and consequently their movement behaviour differs from the main-stream pattern, however we 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 the traffic regulations, and the two assumptions mentioned earlier can be valid only under this prerequisite.

The two assumptions about the movement behavior observed at regulated intersection locations are visually illustrated in Figure 1.7. In Figure 1.7b, two vehicles crossing the same intersection path - turning left at a traffic signal-controlled intersection (intersection B) - exhibit the same movement behavior: slowing down on approaching the intersection (braking indicated as yellow line), indicating the intention to turn left (blinking indicated as red line) and turning left. The same traffic patterns are observed at intersection A, illustrated in Figure 1.7a, where two vehicles show similar movement patterns to each other and to those of intersection B, controlled by the same traffic regulator (traffic lights).


Imagery ©2022 CNES / Airbus, GeoBasis-DE/BKG
Imagery ©2022 CNES / Airbus, GeoBasis-DE/BKG

(a) Intersection A. *(b)* Intersection B.

Figure 1.7.: Both images show intersections controlled by traffic signals. Successive samples during the activation of the turn signal/blinker and brake are indicated in red and yellow respectively. The spatial trajectory of the moving vehicle is indicated in blue.

Other examples illustrating the two assumptions are depicted in Figure 1.8, where the movement paths along with the blinking and braking episodes of two vehicles are shown. The locations indicated in purple represent locations at which all vehicles stop. This behavior may be indicative of a stop sign. The more trajectories that validate a pattern observed at a particular location, the more likely the prediction is to be true. For example, if we assume that forty trajectories rather than just two as in Figure 1.8 validate the stopping behavior, then the stop sign prediction would certainly be more likely. Conversely, if thirty of the trajectories stopped and ten were crossing without slowing down (a mixture of two movement patterns, stop and unhindered driving), then this pattern (mixture of two patterns) could be indicative of a traffic light being there. Moreover, events that are not validated from the majority of the trajectories (black dotted circles) can be ignored as outliers or anomalies (they could be sudden events, traffic anomalies or traffic rule violations).

Therefore, having justified the suitability of GPS trajectories as a data type for determining the type of intersection controls, and having explained the hypotheses for identifying traffic

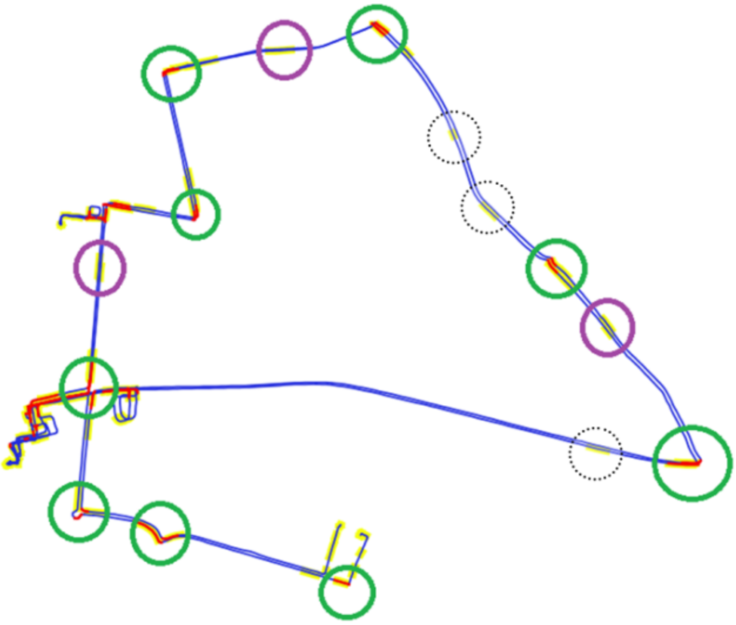


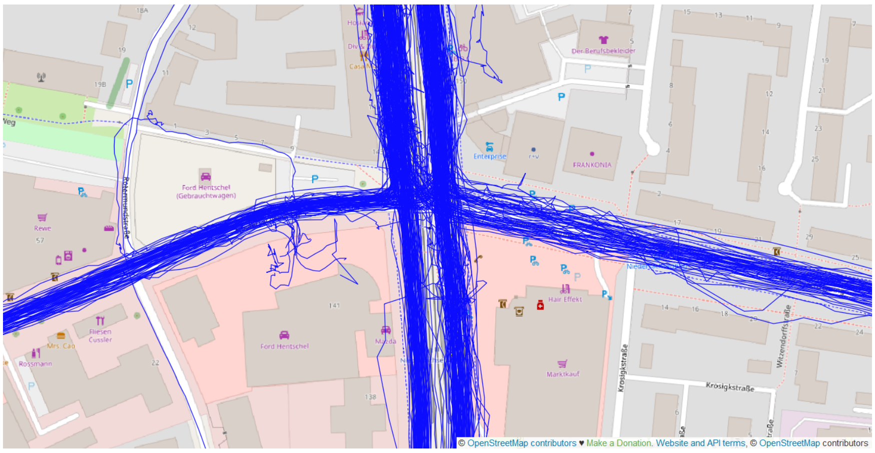
Figure 1.8.: This figure shows the recorded routes of two different vehicles in an area. Blue trajectories denote the routes. Blinker and brake occurrences are shown in red and yellow, respectively. With green and purple circles we denote clusters of similar spatial behavior. Sequences of blinker and brake followed by change of direction indicate turn pattern (green circle) and sequences of braking until stopping indicate stop maneuver (purple circle). Black dotted circles denote brake instances being observed at a single trajectory (not all the drivers that pass from this location brake, so this behavior cannot be considered as pattern).

regulators through the patterns identified in the trajectories, the next step is to validate these hypotheses, which is the subject of this dissertation. The next section lists the research objectives of this thesis.

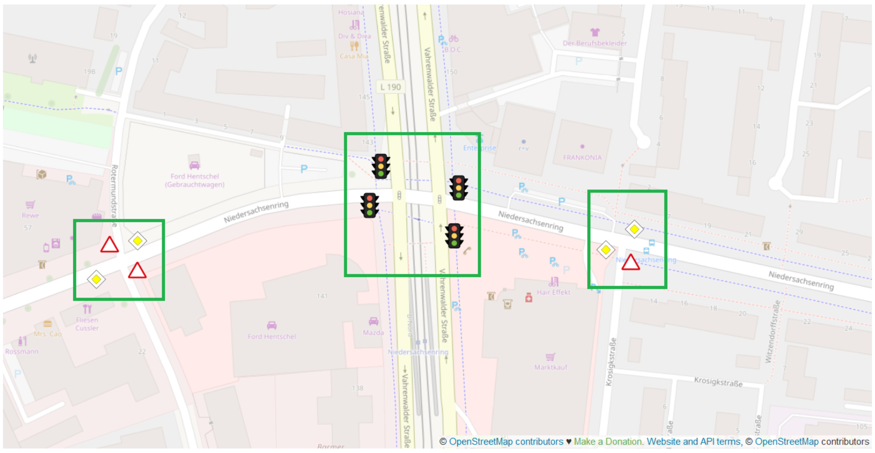
1.5. Research Objectives, Challenges and Contributions

The main research question of this thesis is how traffic regulators can be recognised from low cost data such as GPS trajectories. Schematically, this objective is illustrated in Figure 1.9, which emphasizes also the motivation of this work, that is, as explained in Section 1.2, the enrichment of maps with traffic regulators.

The primary challenge that this thesis had to come up with was the lack of open datasets. Due to this lack, all research works so far use different datasets of various size regarding the number of trajectories and number of intersections that the latter cross, and of various traffic regulations, e.g., only traffic signals, or stop signs and traffic signals, etc. As a result, one can find different proposed methodologies, applied in different datasets, whose results cannot be compared to each other. Moreover, no existing research work so far has been applied to different datasets with different classes of regulators to evaluate the ability of the method to generalize to different classes of rules. It becomes clear that open reference datasets are missing so far, in which both the trajectories and the regulators' ground-truth map (label information) are available and researchers can use them as benchmarks when it comes to compare the classification performance of their proposed approaches with other similar studies. For the scope of this thesis, we manually mapped (and labeled) the regulators of the road network of an open trajectory dataset (Chicago dataset), gathered


Niedersachsenring
Krosigkstraße
Witzendorfstraße
Rotermundstraße
Ford Heritschel (Gebrauchtwagen)
Ford Heritschel
Marktkauf
© OpenStreetMap contributors ♥ Make a Donation. Website and API terms, © OpenStreetMap contributors

(a) Trajectories as data for predicting traffic regulators.


Niedersachsenring
Vahrenwalder Straße
L 190
Krosigkstraße
Witzendorfstraße
Rotermundstraße
Ford Heritschel (Gebrauchtwagen)
Ford Heritschel
Marktkauf
© OpenStreetMap contributors ♥ Make a Donation. Website and API terms, © OpenStreetMap contributors

(b) Traffic regulators as predicted from trajectories.

Figure 1.9.: Schematic illustration of the main objective of this thesis: enriching maps with traffic regulators from low cost data such as GPS trajectories.

trajectories from the city of Hanover and created the groundtruth map of traffic regulators for this dataset too. All these datasets have been placed in an open repository so that other researchers can use to test their methodologies.

Now, having such a benchmark, one can consider whether a method that is good at predicting a certain group of regulators, e.g., traffic lights, stop signs and uncontrolled intersections, can be equally good at predicting another group, e.g., traffic-signals, priority signs, yield signs and uncontrolled intersection; or, whether the same classification features are sufficiently descriptive to distinguish between different groups of regulators; or, whether some existing methods applied in the context of two-class classification problems (traffic light, not traffic light) and achieving excellent performance, are also equally good if applied to multi-class problems.

This thesis addresses the Traffic Regulation Recognition (TRR) problem by testing a new method in different datasets, under different trajectory and classification settings (number of trajectories, one-arm features vs. all- arm features), which to author's knowledge has not been done before. More specifically, the research contributions of the study presented in this dissertation can be summarized as follows:

1. It presents a new methodology for TRR by analysing GPS traces, where in the classification feature vector, information from the adjacent intersection arms is also included (all-arm models).
    - It proposes a modification of a well-known clustering technique for detecting short-term movement events such as stopping and decelerating episodes (Palma et al., 2008).
    - It provides an extensive evaluation of the proposed methodology, which is missing from the literature of the TRR from GPS traces. Three datasets, from different cities (Chicago, Champaign, Hanover), that include different regulation types are used for testing the proposed methodology.
    - It proposes an additional consistency check of the predicted labels at an intersection level, recovering incorrect predicted regulators when possible.
    - It examines whether the sampling rate of GPS tracks influences the classification performance and if so, to what extent.
2. It investigates the classification performance of the proposed TRR method under different trajectory settings.
    - It explores the effect of turning trajectories on the classification performance.
    - It examines the minimum number of trajectories per intersection arm required to achieve acceptable accuracy.
3. It investigates the classification performance of the methodology under sparsely labeled data and streaming data. Labeling data is a time- and cost-consuming task. This thesis explores possible solutions.
    - It explores the predictive ability of the proposed TRR methodology under sparsely labeled data, by using clustering, the semi-supervised learning techniques, self-learning and cluster-then-label, and active-learning.
    - It assesses the cross-city transferabilty of learning, i.e., training the classifier with data from city A and predicting regulators on city B.
    - It tests the proposed methodology under the framework of incremental learning. Instead of processing all data at one time for building a learning model, one observation (data instance) is processed at a time and the learning model is updated incrementally.

1.6. Outline of the Thesis

This thesis is structured as follows:

Chapter 2 provides a brief introduction to the fundamental concepts and methodologies used in this thesis to address its objectives. This theoretical background is mainly concerned with spatio-temporal data, such as GPS tracking and trajectories, and machine learning concepts such as clustering, random forest classification, gradient boosting, self-learning, active-learning, incremental learning and model performance measures.

Chapter 3 discusses relevant existing studies and identifies research gaps in the field of TRR from GPS data.

Chapter 4 explains the datasets used in all the experiments conducted in this thesis, as well as it presents a methodology for TRR using information from a single intersection arm.

Chapter 5 extends the methodology proposed in the previous chapter by including information from all intersection arms (of the same intersection) in the classifier. Here, is also tested the effect of turning trajectories and the number of trajectories on classification performance. A thorough misclassification analysis is also given for all three datasets. Finally, domain knowledge rules are proposed and applied in a post-classification step, where trying to recover the incorrectly predicted traffic regulations (intersection-arm level) taking into account all available predicted regulations at the intersection level (all predicted regulations from all arms that belong to the same intersection).

Section 6 considers the TRR problem under sparsely labeled data. Clustering and semi-supervised techniques are tested with various experimental settings. The transferability of learning between cities (cross-city learning transferability) is also considered here, by training a classifier and use it to predict regulators from a different city than that on which it was trained (i.e. training in city A and predicting in city B). Here the proposed methodology is also extended in the context of incremental learning.

Chapter 7 summarises the findings of previous chapters, underlines the limitations of the proposed methodology and proposes future research directions.

1.7. Summary

Mapping with surveying equipment is a time-consuming and expensive process, which means that it cannot be repeated frequently. The question therefore arises as to how a map and its features can remain up to date. This thesis is motivated by the fact that only a very small amount of intersection regulators is mapped on publicly available maps databases such as OSM. There are four possible areas in which traffic regulators can find applications: 1) fuel-efficient routing, 2) autonomous vehicles, 3) ADAS, and 4) personalized routing. GPS traces are a “lightweight” data source compared to images and can easily be adopted in the context of an opportunistic crowd-sourcing scenario with the scope of mass and fast *sensing* of traffic regulations. This thesis proposes a methodology for recognizing traffic regulators using vehicle trajectories that can be easily recorded with low-cost devices such as mobile phones. Other contributions include the investigation of the problem under sparsely labeled data and the framework of incremental learning.

1.8. Acknowledgements

Chapter 1 contains images, observations, arguments and discussions that have been published in the 3rd AGILE Conference on Geographic Information Science (Zourlidou and Sester, 2015b), in the ISPRS International Journal of Geo-Information,(Zourlidou and Sester, 2019; Zourlidou et al., 2023) and in the 25th AGILE Conference on Artificial Intelligence in the Service of Geospatial Technologies,(Zourlidou et al., 2022d). The thesis author is the principal investigator and author of these articles, as well as the creator of all figures.

2. Theoretical Background

This chapter explains some of the key concepts used in this thesis to achieve its objectives. It clarifies the terms *intersection* and *traffic regulation* (Section 2.1), the main elements that contribute and are related to the acquisition of *movement trajectories* (Section 2.2) and presents some known learning methods of machine learning, such as *clustering*, *classification*, *self-learning*, *active-learning* and *incremental learning* (Section 2.3).

2.1. Intersections and Intersection Traffic Regulations

2.1.1. Intersections

An *intersection*[1] or *at-grade junction* is a junction where two or more paths, roads, highways or other roadways converge, diverge, meet or cross at the same level, as opposed to a *interchange*, which uses bridges or tunnels to separate different roads. Compared to other road structures, intersections have more conflict points. Conflict points are locations within or on the approaches to an intersection where vehicle paths merge, diverge, or cross (Figure 2.1). The more conflict points, the higher the probability of a collision between traffic participants. For highways, the probability of collision between vehicles can be reduced by using interchanges. However, interchanges are not usually feasible for the vast majority of intersections on arterial and collector roads. Certain vehicle movements are also more likely to cause collisions, such as left turns compared to straight and right turn maneuvers. Although intersections represent only a small percentage of the total length of the road network, they nevertheless account for a large proportion of the reported accidents and the majority of potential crash (conflict) locations (Design-Manual, 2021). According to the European Trace project, in the EU-27, about 43% of all road injuries occur at intersections, about 70% of intersection accidents occur within urban areas and 45% to 68% of intersection accidents occur at intersections with traffic signs (Wisch et al., 2019). In addition, the same source reports that in terms of car accidents with at least serious injuries, 38% of the 34,489 car-to-car accidents occurred at intersections, accounting for 18% of deaths (a total of 4,236 deaths).

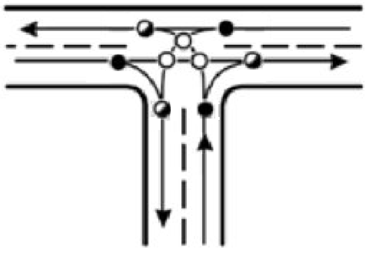
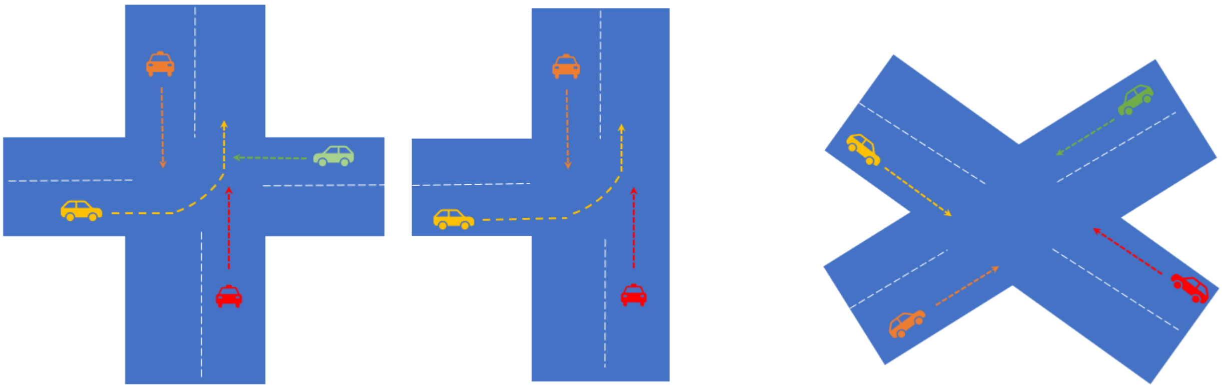


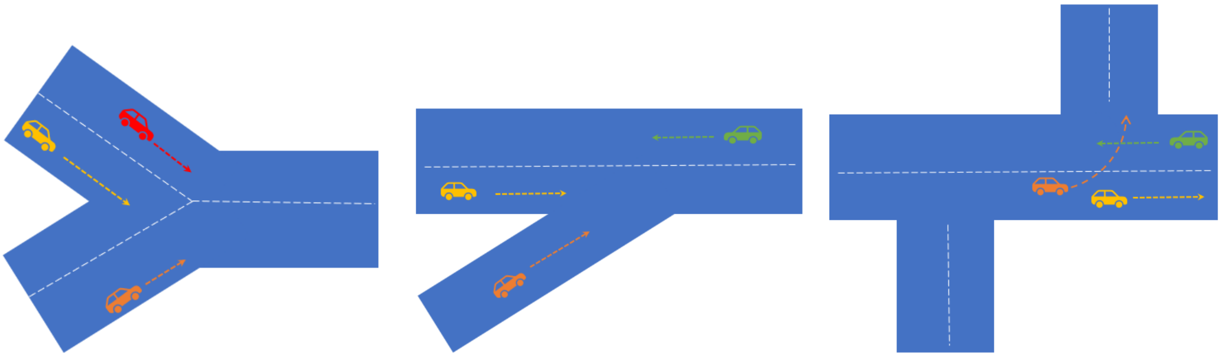
Figure 2.1.: Vehicle conflict points at a T-intersection: ◐ merging, ○ crossing, ● and diverging points.

---

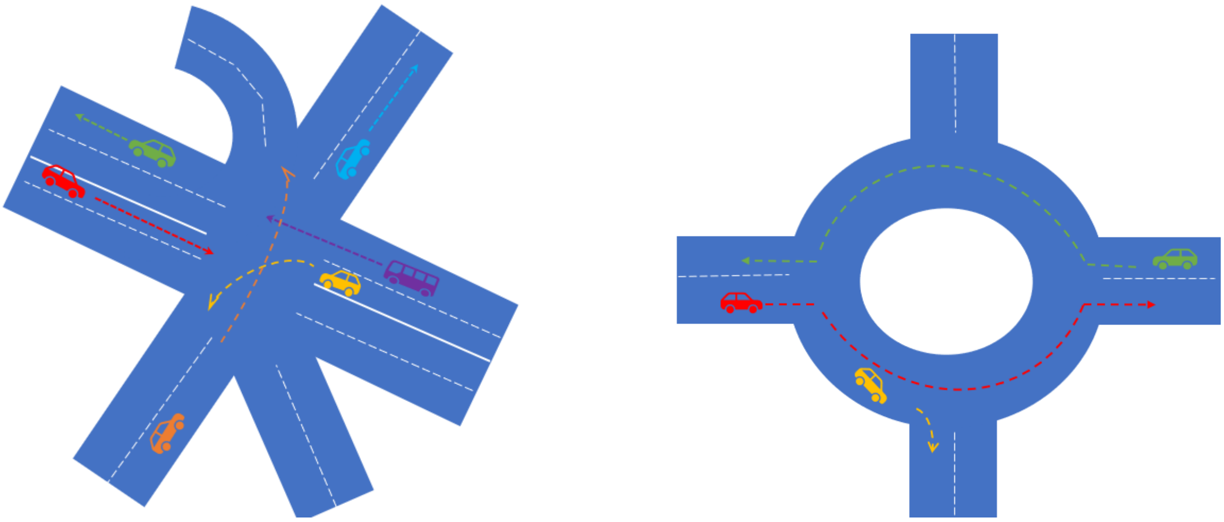
[1] The term *crossroad* is often used to describe the same concept. However, an intersection refers primarily to a road that joins two main roads or crosses a main road.



(a) Crossroad (four-leg). *(b)* T-intersection (three-leg). *(c)* X-intersection (four-leg).



(d) Y-intersection (three-leg). *(e)* Ramp merge (three-leg). *(f)* Misaligned intersection (four-leg or offset-right).



(g) Deformed intersection (multi-leg). *(h)* Roundabout.

Figure 2.2.: Different types of intersections.

The intersections are crossed by different types of participants, such as vehicles, motorcycles, bicycles and pedestrians. Based on their topology, Wei et al. (2021) classifies intersections into eight main categories: *crossroad*, *X-intersection*, *Y-intersection*, *T-intersection* (with variations in angle of approach), *roundabout*, *misaligned intersection*, *ramp merge*, and *deformed intersection* (Figure 2.2). Other simpler categorization may be according to the number of *arms* or *legs* or *approaches* they have (three-leg intersection, four-leg intersection, multi-leg intersection, as illustrated in Figure 2.2). The X-intersection is also found under the name *scissor* or *skewed cross*. In general, each intersection can vary considerably in terms of the scope, shape and type of traffic control devices. The simplest and most common T-intersection is the private entrance or driveway where access is given to a residential property to and from the road network, while at the other extreme, a highway intersecting another major highway usually requires a rather complex intersection design. The factors taken into consideration for the design of an intersection are listed in Table 2.1.

Table 2.1.: Intersection design considerations (Design-Manual, 2021).

Human factors		
Driving habits	Driver error	Driver workload
Driver expectancy	Driver distractions	Pedestrian use and habits
Conformance to natural paths of movement	Perception-reaction time	Bicycle traffic use and habits
Visual recognition of roadway cues	Compatibility with context characteristics	Demand for alternative mode choices
Traffic Considerations		
Design users, modal priority and intersection design vehicles	Design and actual capacities	Design-hour turning movements
Variety of movements (diverging/merging/weaving/crossing)	Vehicle size and operating characteristics	Vehicle speeds
Transit involvement	Crash Experience	Bicycle movements
Pedestrian movements		
Physical Elements		
Character and use of abutting property	Vertical alignments at the intersection	Sight distance
Angle of the intersection	Conflict areas	Speed-change lanes
Managed lanes (HOV, HOT, shoulder)	Accessible facilities	Parking zones
Geometric design features	Traffic control devices	Illumination
Roadside design features	Environmental factors	Crosswalks
Transit facilities	Driveways	Streetside design features
Adjacent at-grade rail crossing	Access management treatments including turn restrictions	
Economic Factors		
Cost of improvements	Annual maintenance	Annual operation costs
Annual life cycle costs	Salvage value	Energy consumption
Emissions		
Effects of controlling access and right of way on abutting properties where channelization restricts or prohibits vehicular movements		

As explained later on, the methodology proposed in this thesis is based on the assumption that certain *physical elements* of the road infrastructure, such as the length of the road connecting an intersection to the nearby ones, the speed limit, the category of the road (primary, secondary, etc.), as well as traffic-related elements such as the observed speed of vehicles when crossing intersections and certain movement patterns (stopping and slowing episodes) can be indicative of the traffic regulations that control the intersections. That is, since intersections are designed to serve both traffic and traffic participants in a particular way (Table 2.1), their design (including the regulations by which they are controlled) both influences and is influenced by the traffic and the movement behavior of traffic participants. Therefore, the methodology of this thesis assumes that by exploiting infrastructure and movement characteristics, such as those mentioned previously, extracted from open maps and GPS traces, the traffic regulations of intersections, as attributes of the design of intersections, can be recovered.

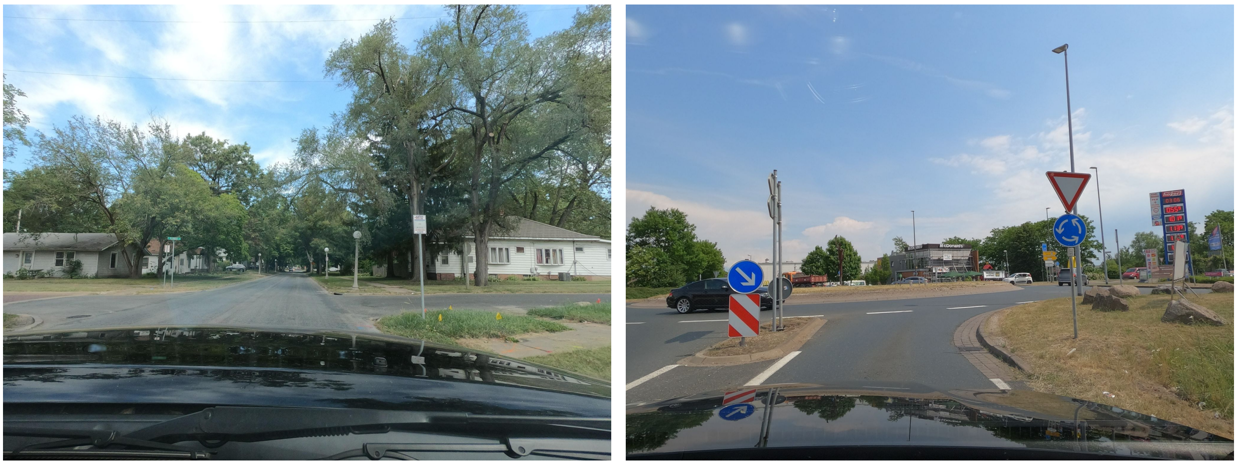
2.1.2. Intersection Traffic Regulations

Traffic regulations are all applicable laws concerning the use and operation of vehicles. Traffic participants in Germany are subject to the government's Road Traffic Code (Straßenverkehrsordnung, StVO), which is a set of road traffic rules that every participant must adhere to. In the context of this thesis, this term refers only to a subset of traffic regulations, that regulates traffic at intersections. Common types of intersection control include *uncontrolled intersection* (right of way rule), *yield signs*, *priority signs*, *stop signs*, *roundabouts*, and *traffic control signals* (traffic lights). The description of the intersection controls that follows is according to the Design-Manual (2021).

Uncontrolled intersections (Figure 2.15a) do not have signaling and the right of way rule applies. This type of intersection is usually found on local roads where the volume of intersecting roads is low and roughly equal, speeds are low and there is little history of accidents.

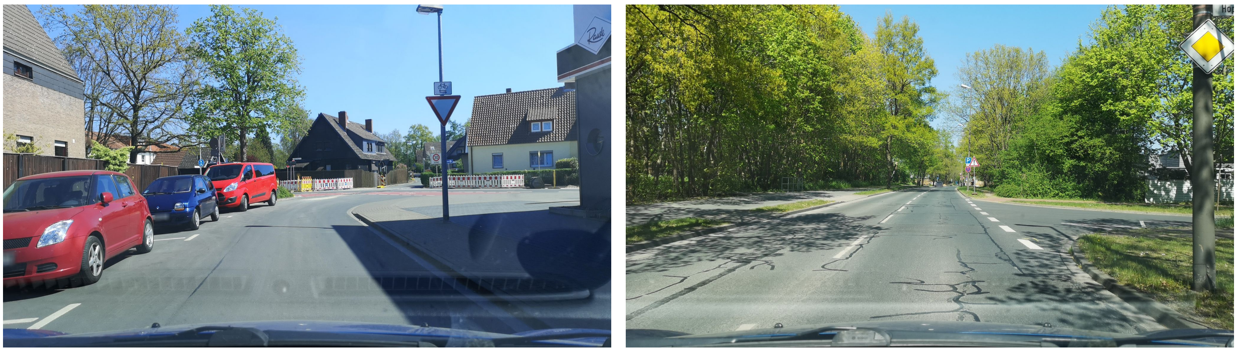
Intersections with yield control (Figure 2.3c) give priority to other intersecting roads without requiring a stop, i.e. if there is no other vehicle to give priority, the driver does not need to stop. This control is mainly used at three-way intersections with low traffic volumes. The yield control is generally not recommended when pedestrians are expected. In the US it is most commonly used in rural areas and is not recommended in urban locations. In contrast, in Germany it is used very often and in urban locations, which is combined with priority control (Figure 2.3d) on the other legs of the same intersection. For example, at a T intersection, if one approach (leg) of the intersection is yield controlled (yield traffic sign), the other two approaches are priority controlled (priority traffic sign).

Two-way stop control intersections are a common, lower cost control that requires traffic on the minor road to stop and yield to mainline before entering the major road. Multi-way stop control (Figure 2.3e) requires all traffic to stop before entering the intersection. It is suitable for lower speed facilities with approximately equal volumes on all legs. Multi-way stop control is not recommended for multi-lane routes or intersections with unbalanced directional traffic flows due to the delays and queues introduced on the major-volume legs of the intersection. Compared to two-way stop controls, they have fewer fatal and injury accidents. In addition, traffic delays at these locations are increased, as well as fuel consumption and air pollution. In Germany it is common for stop signs to coexist with priority signs at the same intersection, i.e., at a T-junction, one leg is controlled by a stop sign and the other two by priority signs. In the USA, two-way stop and all-way stop controls are more common.



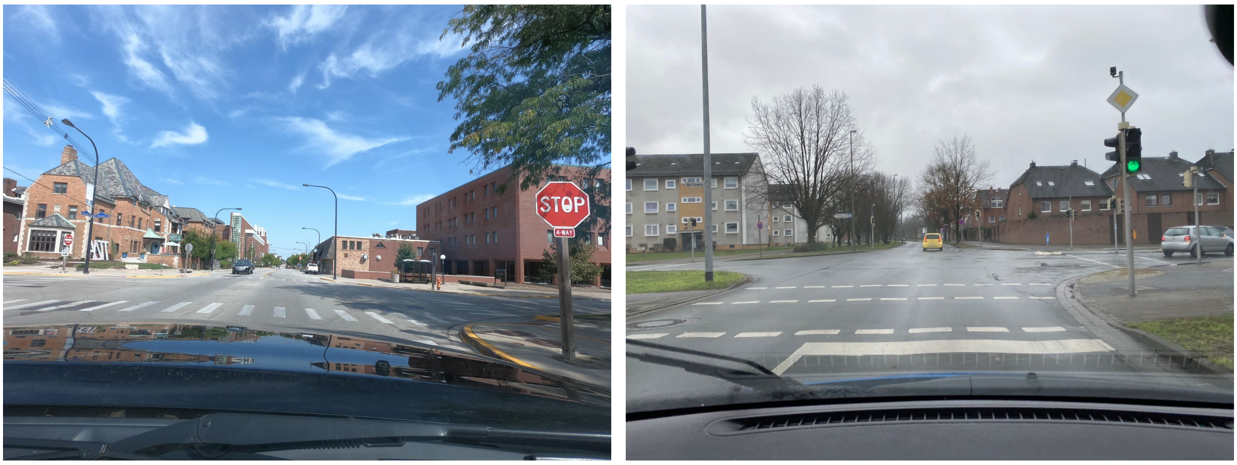
(a) Uncontrolled T-intersection (Four-leg inter.).

(b) Roundabout.



(c) Yield controlled intersection leg (T-inter.).

(d) Priority controlled intersection leg (T-inter.).



(e) All-way stop controlled intersection (Four-leg inter.).

(f) Traffic signal controlled intersection (Four-leg inter.).

Figure 2.3.: Images from intersections controlled by different regulations[2].

Roundabouts (Figure 2.15b) are often circular (or near-circular) at-grade intersections, where traffic on the approaches gives priority to traffic within the circulating roadway. Among the various advantages they have as intersection controls are fewer points of conflict, reduced fatal and injury crashes compared to other types of at-grade intersections,

---

[2] Image Sources: `https://www.mapillary.com/app/`.

reduced traffic delays, greater capacity from a two-way or multi-way stop, and aesthetic treatments and gateways to communities.

Intersections controlled by traffic signals (traffic lights) are called *signalized* (Figure 2.3f). Compared to stop controls, traffic control signals increase the capacity of the intersection and reduce at-angle vehicle accidents. They are often used to provide priority service in railroads, emergency responders, transit, and approaches where advance queue loops are used. They can also be used to stop heavy traffic at intervals to allow other traffic, vehicular or pedestrian, to complete its movement or enter the intersection. However, signalized intersections require continual maintenance and engineering for optimal operation and can be susceptible to power outages and detection failures. Also, rear-end accidents occur more frequently at signalized intersections compared to other type of control intersections. An additional disadvantage is that they cannot adequately balance high traffic flows with pedestrian demands.

2.2. Spatiotemporal Data and Movement Trajectories

This section explains some main concepts related to movement trajectories, such as the Global Positioning System (GPS) (Paragraph 2.2.1), the sampling frequency (Paragraph 2.2.2), the GPS exchange format (Paragraph 2.2.3) and the movement patterns in spatiotemporal data (Paragraph 2.2.4).

2.2.1. The Global Positioning System

GPS is a space-based radio navigation system, developed by the US Department of Defense in the early 1970s for military purposes. Later, it was made available to civilians and is now accessible to both military and civilians. Its main purpose is to provide continuous positioning and timing information anywhere in the world in all weather conditions. Because of the numerous users it serves, as well as for security reasons, GPS is an one-way (passive) system. It is operated and maintained by the US Department of Defense.

GPS consists of a constellation of satellites[3], which broadcast navigation signals and a network of ground stations and satellite control stations used for monitoring and control. Currently, 31 GPS satellites orbit the Earth at an altitude of about 17,700 km. The three segments of GPS are the space segment, the control segment and the user segment and are illustrated in Figure 2.4. Each GPS satellite transmits a signal containing two sine waves (known as carrier frequencies), two digital codes and a navigation message. The carriers and codes are used to determine the distance of the user's receiver from the GPS satellites. The navigation message contains the coordinates of the satellites (position) as a function of time, together with other information. These signals are controlled by high-accurate atomic clocks located on the satellites.

The control segment consists of a global network of monitoring stations and ground control stations, with a main control station located in Colorado Springs, Colorado, US. Its main task is to track the satellites in order to determine and predict their location, to maintain the system integrity, to track the atomic clocks and atmospheric data, etc. This information

---

[3] According to the US Federal Aviation Administration, there are 31 satellites in the GPS constellation, 27 of which are in use at any given time, while the rest are available as backup. `https://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/navservices/gnss/gps`. Accessed 17.08.2022.

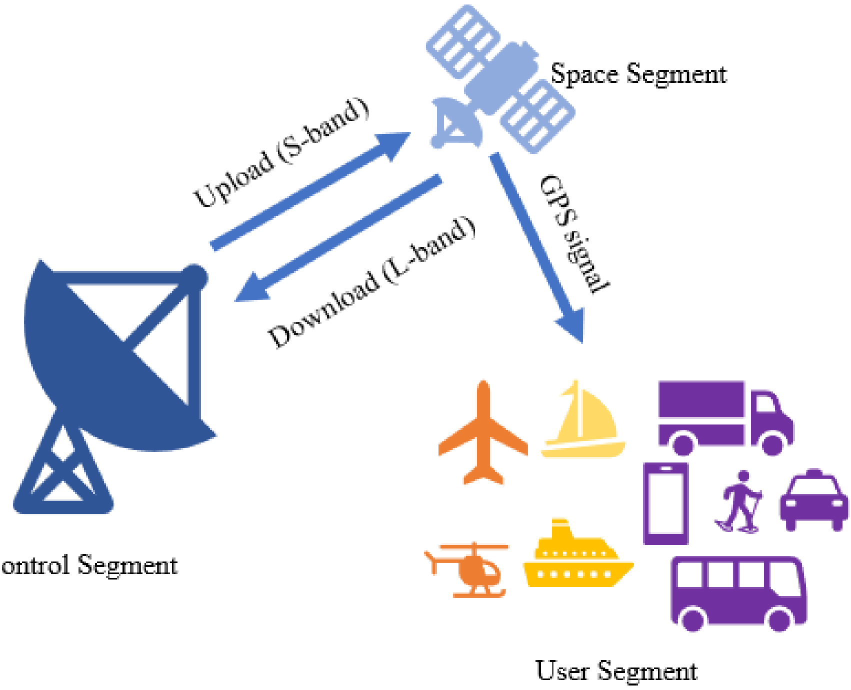


Figure 2.4.: The GPS segments.

is packed and uploaded to the satellites via the S-band link (Figure 2.4). The user segments include all users who, with a GPS receiver connected to a GPS antenna, can receive GPS signals and therefore determine their position anywhere in the world.

The idea behind GPS is rather simple. If the distances from a point on Earth (GPS receiver) to three GPS satellites, as well as the positions of the satellites, are known, then the 3D position of the GPS receiver can be calculated by applying the well-known concept of *position resection*. The receiver uses the time difference between the time of signal reception and the time of transmission to calculate the distance, or range, from the receiver to the satellite. To calculate the distances from these three signals, an atomic clock synchronized with the GPS is required. However, by taking a measurement from a fourth satellite, the receiver avoids the need for an atomic clock. Thus, the receiver uses four satellites to calculate latitude, longitude, altitude and time. The basic GPS service provides users with an accuracy of about 7.0 meters, at 95 percent of the time, anywhere on or near the earth's surface (U.S. Federal Aviation Administration, 2022).

GPS signals are affected by various types of random errors and bias (systematic errors), which can be classified into those originating from the satellites (orbital errors, satellite clock errors), at the receiver (clock errors, multipath errors, receiver noise, antenna phase centre variations) and those caused by signal propagation (atmospheric refraction - the GPS signal is delayed as it passes through the ionospheric and tropospheric layers of the atmosphere). In addition, the geometric positions of GPS satellites as seen by receivers also affect the accuracy of the estimated GPS position. The more dispersed the satellites in the sky are, the better the accuracy obtained (El-Rabbany, 2002, p. 27).

Other satellite constellations also provide similar services. Collectively, these constellations and their extensions are called Global Navigation Satellite Systems (GNSS). The GLONASS satellite system is a global navigation satellite system developed and operated by Russia, consisting of 24 satellites[4] in orbit (22 operational) at a nominal altitude of 19,100 km. The BeiDou navigation system is developed and operated by China. It has 35 satellites in orbit[5] including 5 satellites in geostationary orbit (GEO) and 30 non-GEO satellites. The system has evolved from a regional system, called BeiDou-1, to a global navigation satellite system after the last launch on 23 June 2020, offering global coverage for timing and navigation. The Galileo global satellite is developed and operated by the European Union (first launch in 2011) and consists of a constellation of 30 satellites (including 6 spare satellites)[6]. All providers have offered free use of their respective systems to the international community and have developed International Civil Aviation Organization (ICAO) Standards and Recommended Practices to support the use of these constellations for aviation (U.S. Federal Aviation Administration, 2022).

2.2.2. Sampling Frequency

Most GPS receivers calculate the position and velocity every second, i.e., one sample per second. Depending on the application, the sampling rate can be adjusted accordingly. For example, for a taxi tracking management application, a low sampling rate (less than 1 Hz) may be sufficient for the tracking task and efficient in terms of storage/network demands. Andersen and Torp (2017), investigating the effects of sampling rate on trajectory paths and travel time on the road network, found that GPS data collected every sixty seconds is inaccurate to use for calculating travel times and that trajectory-based map matching works best if the sampling period is twenty seconds or less.

The sampling period T is defined as the time difference between two consecutive GPS samples ($T = t_i - t_{i-1}$). However, if for any reason there is a temporal gap between two consecutive positions, i.e. $t_i - t_{i-1} > T$, it means that sampling has been interrupted. Such a gap in the track is called a *hole* (Parent et al., 2013). Nevertheless, if the gap was caused intentionally (e.g., an employee stops her GPS when she goes to lunch), then the temporal gap is called *semantic gap*. Holes can generally be recovered using interpolation techniques, while semantic gaps should not be recovered, as they are intentionally omitted (Parent et al., 2013). A common reason for interrupted sampling (holes) is when not enough signals are received from the satellites (GPS signal loss). Many GPS devices ideally need to receive signals from at least seven to eight satellites to calculate their position to an accuracy of about 10 meters. With fewer satellites, uncertainty and inaccuracy increase, and with fewer than four satellites, many receivers have difficulty producing reliable position estimates and report that the GPS signal has been lost.

Figure 2.5 illustrates two trajectories (blue and orange) of two moving objects that traversed the same movement path and their position was sampled at different sampling periods. Each dot represents the position of a moving object in space at a particular time. By connecting the successive positions (in time order), we obtain the trajectories. The blue trajectory represents the path traveled, where the position (blue dots) is sampled every T time units, while the orange trajectory is sampled every $3T$ time units. The orange trajectory can be seen as the result of *down sampling* (or undersampling) the recorded (blue) locations, obtaining a *less informative* representation of the path. Intermediate locations lost due to

---

[4] Source: `https://gssc.esa.int/navipedia/index.php/GLONASS_General_Introduction`. Accessed 17.08.2022.

[5] Source:`https://gssc.esa.int/navipedia/index.php/BeiDou_Space_Segment`. Accessed 17.08.2022.

[6] Source:`https://gssc.esa.int/navipedia/index.php/Galileo_Architecture` Accessed 17.08.2022.

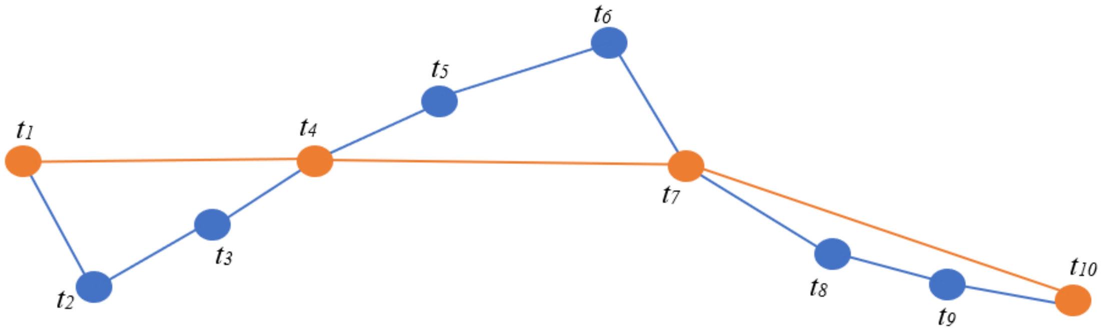


Figure 2.5.: Two trajectories that represent the same traveled path, sampled with different sampling periods (blue with T period and orange with 3T period).

undersampling (e.g., locations at t_2, t_3, t_5, t_6, t_8, t_9) cannot be recovered and are therefore not expressed in the orange trajectory. Formally a trajectory can be defined as following:

Definition 1. (Trajectory sample) A *trajectory sample* is a list of space-time points

$$\{p_0 = (x_0, y_0, t_0), p_1 = (x_1, y_1, t_1), ..., p_N = (x_N, y_N, t_N)\},$$

where space points are indicated as $x_i\ y_i \in \mathbb{R}$, $t_i \in \mathbb{R}^+$, for $i = 0, 1, ..., N$, and time instances as $t_0 < t_1 < t_2 < \ ... \ < t_N$. ■

Open GPS trajectory datasets contain data sampled at various sampling rates. The Geolife dataset (Zheng et al., 2011) contains 17,621 trajectories from different modes of travel (car, bus, walking, subway, bike and train) recorded by different GPS loggers and GPS-phones with different sampling rates. 91% of the trajectories are recorded in a relatively dense representation, i.e., every 1-5 seconds or every 5-10 meters per point. The one month Beijing taxi GPS trajectory dataset (Lian and Zhang, 2018) recorded in May 2009 contains 129 million data samples, with 75% of the data sampled in one-minute intervals.

2.2.3. The GPS Exchange Format: GPX

The GPS Exchange Format, GPX, is a lightweight XML data format used to exchange GPS data between GPS receivers, desktop and mobile software and web-based services on the Internet, with a growing list of programs for Windows, MacOS, Linux, Palm, and PocketPC. Although simple, it is powerful enough to describe complex geographic objects. It can store three types of data: *waypoints*, *tracks*, and *routes* (Figure 2.6).

A waypoint (complex type: wptType) is a single point consisting of the WGS 84 (GPS) coordinates of a point and possibly other descriptive information. It represents a waypoint, a point of interest or a named feature on a map. A route (complex type: rteType) is an ordered list of waypoints (routpoints) representing a series of turnpoints leading to a destination. In a very simple GPS navigation device, this might be a straight line from the starting point to the destination. In a modern device, it is the route leading from the starting point to the destination. In a route, waypoints ensure that a particular route is followed. However, waypoints can allow GPS navigators to adjust the route in the event of a detour, meaning that if the user misses a waypoint, the navigator will recalculate the route and either direct the user to the next waypoint from the one missed or add another waypoint if necessary to reach the destination. A track (complex type type: trkType) is an ordered list of points describing a path. A track consists of a sufficient number of points (trackpoints) to accurately represent each bend of a path. Compared to routes, tracks are much more detailed representations of paths and also work differently in that they are an accurate record of the path taken, and therefore, although they can be navigated, they do not allow the GPS navigator to recalculate a route.

Technically, since GPX is based on XML, it inherits all the advantages of XML. XML is an open standard, with a rapidly growing base of developers and tool providers. GPX defines a common set of data tags for describing GPS and geographic data in XML (Figure 2.7) allowing developers to define their own private objects and attributes. GPX files can be converted to other file formats, such as .KML, .KMZ (Google Earth), .CSV (Microsoft Excel), GEOJSON and .shp, using a simple web page or conversion program. A list of software applications and websites that support the GPX format can be found on the official GPX website (topografix.com, 2022).

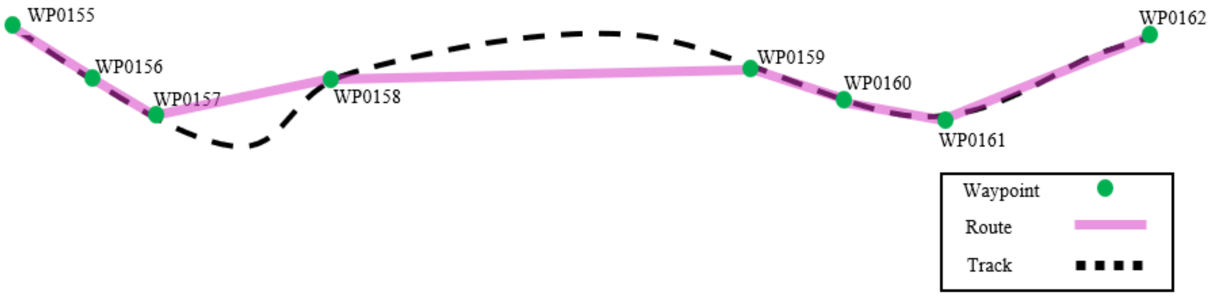
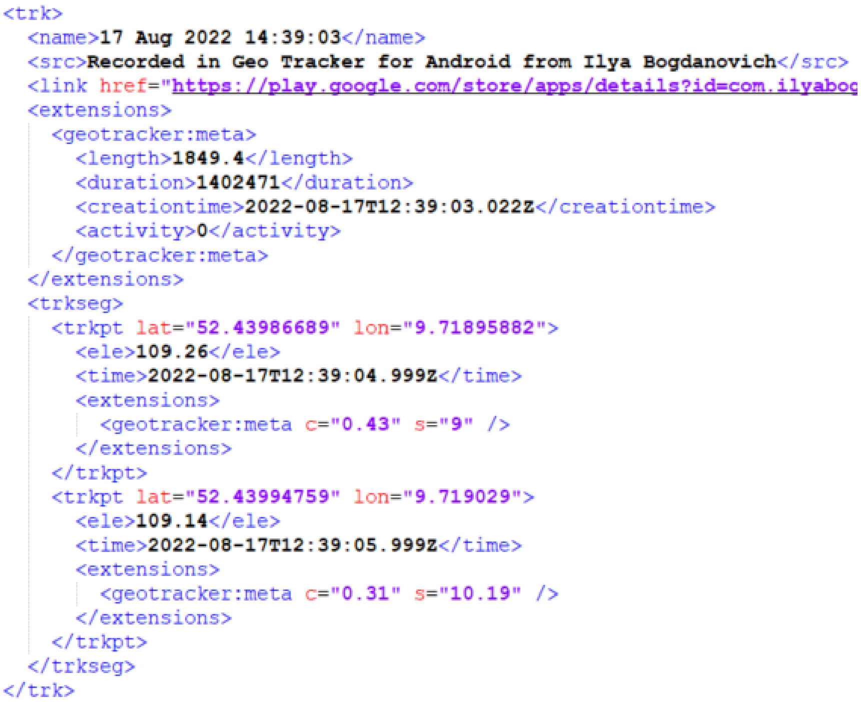


Figure 2.6.: Type of GPS data that can be stored in a GPX file: waypoints, routes and tracks.



```xml
<trk>
  <name>17 Aug 2022 14:39:03</name>
  <src>Recorded in Geo Tracker for Android from Ilya Bogdanovich</src>
  <link href="https://play.google.com/store/apps/details?id=com.ilyabog
  <extensions>
    <geotracker:meta>
      <length>1849.4</length>
      <duration>1402471</duration>
      <creationtime>2022-08-17T12:39:03.022Z</creationtime>
      <activity>0</activity>
    </geotracker:meta>
  </extensions>
  <trkseg>
    <trkpt lat="52.43986689" lon="9.71895882">
      <ele>109.26</ele>
      <time>2022-08-17T12:39:04.999Z</time>
      <extensions>
        <geotracker:meta c="0.43" s="9" />
      </extensions>
    </trkpt>
    <trkpt lat="52.43994759" lon="9.719029">
      <ele>109.14</ele>
      <time>2022-08-17T12:39:05.999Z</time>
      <extensions>
        <geotracker:meta c="0.31" s="10.19" />
      </extensions>
    </trkpt>
  </trkseg>
</trk>
```

Figure 2.7.: A sample of the content of a GPX file, containing a track consisted of two trackpoints, with sampling period of 1 second.

2.2.4. Movement Patterns in Spatiotemporal Data

Spatiotemporal data is any information related to space and time (Gudmundsson et al., 2008). In this context, the representation of temporal sequences of spatiotemporal positions, i.e., pairs of time-position of a moving object[7] is called *trajectory*. Depending on the recording device, additional data, such as velocity and direction, may be attached to each time-position pair. Here we adopt the definition of Giannotti et al. (2007) for the trajectory.

Definition 2. (ST-sequence) A s*patio-temporal sequence* (ST-sequence) or *trajectory* is a sequence of triplets $S = \langle (x_0,y_0,t_0),...(x_k,y_k,t_k) \rangle$, where t_i with $i \in \mathbb{N}^0$, is a timestamp, $\forall k : 0 \leq i \leq k$, $t_i < t_{i+1}$ and (x_i,y_i) are points in $\mathbb{R}^2$. ■

Movement patterns are concise descriptions of frequent behaviors, both in terms of space (i.e., locations visited by a moving object) and time (i.e., the duration of the movement behavior). A movement pattern can be *any* frequent behavior that can be modeled as any arrangement of sub-trajectories and can be defined and formalized sufficiently, so that it can eventually be recognized by an algorithm, as only formalized patterns are detectable by algorithms (Gudmundsson et al., 2008). For example, suppose we want to define a pattern that represents a set of individual trajectories that share the property of visiting the same sequence of locations with similar travel times. Such a pattern could be used in a touristic scenario, as it could reveal sequences of places visited by tourists, and hence could be used to suggest personalized services (transport ticket offers, food vouchers, etc.). Such a pattern can be defined as follows (Giannotti et al., 2007):

Definition 3. (T-pattern) A *trajectory pattern* or *T-pattern*, is a pair (S,A), where $S = \langle (x_0,y_0),...,(x_k,y_k) \rangle$ is a sequence of points in $\mathbb{R}^2$, and $A = \langle \alpha_1,...,\alpha_k \rangle \in \mathbb{R}^k_+$ is the (temporal) annotation of the sequence. T-patterns can also be represented as $(S,A) = (x_0,y_0) \xrightarrow{\alpha_1} (x_1,y_1) \xrightarrow{\alpha_2} ... \xrightarrow{\alpha_k} (x_k,y_k)$. ■

In this particular example of a trajectory pattern, the occurrence of a T-pattern occurs when both the spatial positions and transition times of the pattern correspond to those found in an input sequence (trajectory) being searched for a T-pattern match.

The common element of the trajectories that make up a pattern is a set of distinguishing characteristics that form a concise description of the set of trajectories. These summary descriptions are called patterns or behaviors by Laube (2009). For trajectory behavior, we adopt the definition of Parent et al. (2013).

Definition 4. (Trajectory behaviour) A *trajectory behaviour* (behaviour, for short) is a set of characteristics that specifies a peculiar bearing of a moving object or set of moving objects. When this set of characteristics is observed on an individual trajectory, it is called *individual trajectory behaviour*. Otherwise, when the behaviour applies to a non-empty set of trajectories it is called *collective trajectory behaviour*. ■

Recalling the tourist scenario mentioned earlier, a tourist behavior can be a daily trajectory that exhibits this behavior if: its starting point $P1$ is a place of the "accommodation" type (e.g., a hotel), it makes at least one stop at a place of the "museum" or "tourist attraction" type, it makes one stop at a place of the "food" type, and its ending point is in the same place as the starting point $P1$. The set of characteristics that determines tourist behaviour

---

[7] A moving object is also called *entity*.

is the “type of places”, associated with the starting point, the ending point and the stops of the trajectory, and possibly the order in which tourists visit them.

Therefore, movement patterns in spatiotemporal data refer to (usually salient) events and episodes represented in a set of trajectories, as expressed by a set of moving objects. According to this definition, a pattern can be seen as an aggregate representation or abstraction of many individual trajectories of moving objects from an observed population of the latter. It should be noted that the individual trajectories that “participate” in a pattern do not necessarily express movements that take place at the same period of time (in the same place and time).

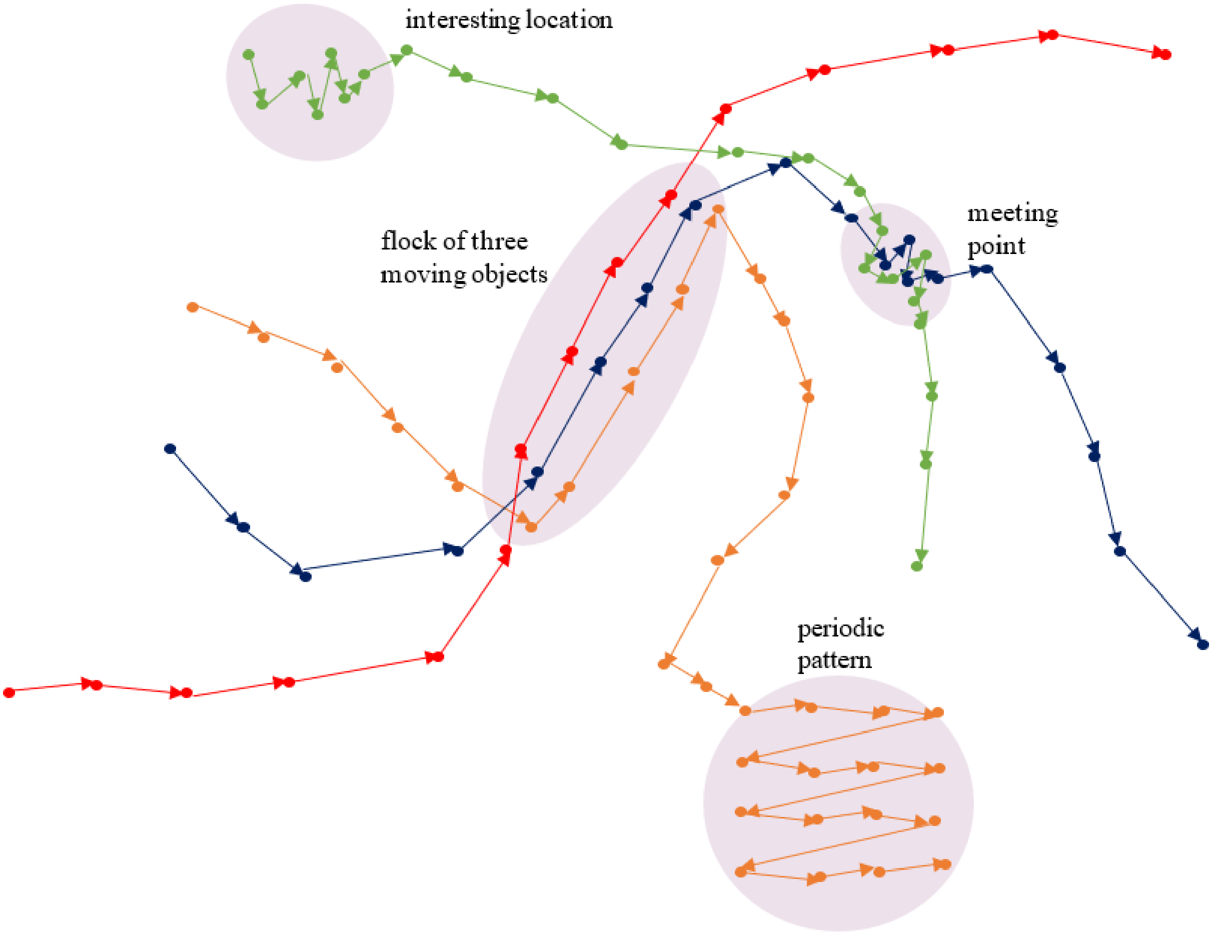


Figure 2.8.: Four movement patterns identified in the spatiotemporal data (trajectories) of four moving objects: interesting location, meeting point, moving clusters (flock) and periodic behaviour.

Figure 2.8 illustrates four different movement patterns in the trajectories of four entities. An *interesting* location (Comito et al., 2016), also known as *movement attractor* or *Region of Interest (*RoI) (Giannotti et al., 2007) or *significant* location (Niu et al., 2021) is defined as the pattern of a single object (in this example) or that observed in several objects remaining in a particular location for a given period of time. A RoI represents a natural way of partitioning space into meaningful regions and then using them to associate trajectories of moving objects within them. An example of a RoI can be a shopping mall (of interest to many entities) or an entity’s residence (of interest to a particular individual). A *meeting point* is a location where moving objects are present at the same time period. When a spatiotemporal pattern repeats with some periodicity, it is called *periodic*, e.g., the journey from home to work every weekday morning. A cluster of entities moving simultaneously is called *flock*. Ignoring the time of movement development of the three entities in the example of flock shown in Figure 2.8, such a spatial coexistence is usually referred as a *cluster of trajectories*. From these examples it is clear that a movement pattern usually involves a certain number of entities and trajectories, may start and end at certain times (temporal footprint) and/or be confined to a certain space (spatial footprint) (Gudmundsson et al., 2008). However, patterns can also be seen as repetitive movements, observed in the motion of individual moving objects without necessarily being associated with a specific location or (unique) behaviour (e.g., an episode of deceleration of a moving object).

The practicality of detecting movement patterns can be demonstrated through the various applications that can use them. In the context of traffic management, an example of a traffic pattern is the traffic jams at major city intersections during rush hours or driving at low speed at places where traffic calming devices are installed. In the context of *animal behaviour* or *ecology*, movement patterns may be the different phases of the annual cycle of migratory animals or the sequential use of habitats by other animals (De Groeve et al., 2016). In a *surveillance* or *public safety* scenario, where human movements are monitored by cameras, normal movement patterns and suspicious (abnormal) behaviours can be identified (Makris and Ellis, 2005), to prevent crime. In a *military scenario*, moving object databases allow for the dynamic updating of the location of soldiers and military vehicles and can identify complex movement patterns, such as the current convergence area where the enemy is massing his troops. Finally, in *professional sports*, such as football or tennis, coaching teams analyse the movement behaviour of their own and the opposing team (sports scene analysis) to identify patterns such as player-ball interactions and strategies (Feuerhake, 2016). The next paragraph presents the main techniques for spatiotemporal data mining.

2.2.5. Spatiotemporal Data Mining

From the above examples of movement patterns it is clear that the possibilities for defining movement patterns are numerous. For this reason there are numerous data mining algorithms for discovering new, non-trivial but potentially useful patterns in (large-scale) spatiotemporal data. According to a recent survey on spatiotemporal data mining techniques (Sharma et al., 2022), there are six major families of patterns, and therefore equal categories of data mining: 1. *spatiotemporal outliers and anomalies*, 2. *spatiotemporal couplings and tele-couplings*, 3. *spatiotemporal prediction*, 4. *spatiotemporal partitioning and summarization*, 5. *spatiotemporal hotspots*, and 6. *spatiotemporal change*. *Outliers* are observations that appear to deviate from the expected or collective behavior that is in the spatial domain joined to an exact position (x,y) or a location type, e.g. road bump. A spatial outlier is a spatially referenced object whose spatial or non-spatial attribute value differs significantly from other spatially referenced objects observed in the same spatial neighborhood. For example, in a surveillance scenario of a public space, a deviation from the common pedestrian path that individuals typically walk on may raise an alarm about possible illegal activities in the environment (e.g., building intrusion). The detection of spatial time series outliers is usually based on visualization approaches such as variograms and Moran scatter plots or based on capturing exceptional cases that deviate substantially from the majority of patterns (clusters). When searching for patterns in Big Data, parallel computing techniques, such as tile processing, are used to efficiently manage the computational load (multiple tiles are processed simultaneously using multi-threading). Techniques for detecting outliers in trajectories include summarization of patterns (Lam, 2016), common subsequences (He et al., 2022) and trajectory scores (Abreu et al., 2021), among others. For example, in the latter work, a visual analytics tool that uses spatial segmentation, divides trips into subtrajectories and score them. These scores are displayed in a tabular visualization where users can rank trips by segment to find local anomalies. The amount of interpolation in subtrajectories is displayed together with scores so that users can use both their insight and the trip displayed on the map to determine if the score is reliable.

Instances that occur close in space and time are called *spatiotemporal coupling patterns*. When these patterns are unordered they are called *co-occurrences*, when they are partially ordered they are called *cascading patterns*, and when they are ordered they are called *sequential patterns*. An example of a partially ordered cascading pattern is the finding that bar closings lead to drunk driving and assault. *Tele-coupling patterns* identify a significant positive or negative correlation in a spatial time series. Common techniques for extracting co-occurrences include a-priori-based methods and mixed-drove co-occurrence pattern finding (Andrzejewski and Boinski, 2021), for sequence patterns include spatial participation sequence indices (Maciag et al., 2019) and probabilistic approaches and for tele-coupling patterns include spatial autocorrelation and statistical significance techniques (Kawale et al., 2012).

Spatiotemporal predictions predict a target variable from a set of forecast-dependent variables. The predicted variable can be continuous or discrete and either spatiotemporal regression (Harris et al., 2017) or classification (Yuan and Li, 2021) can be used for predicting it.

Spatiotemporal partitioning or *clustering* is the process of clustering similar spatiotemporal data and thus partitioning the underlying space and time. For example, crime data, which is spatial and temporal in nature, can be partitioned to help law enforcement agencies identify crime trends and effectively deploy their police resources. *Spatiotemporal summarization* provides a compact representation of spatiotemporal data. For example, road accident events can be summarized into major routes covering most accidents. Spatiotemporal summarization is often done after or along with the spatiotemporal partitioning, so that the objects in each partition can be summarized with aggregate statistics or representative measures (Shekhar et al., 2015). Common approaches for spatiotemporal partitioning of events are density-based methods such as DBSCAN (Ester et al., 1996) and ST-DBSCAN (Birant and Kut, 2007), hierarchical methods such as BIRCH (Zhang et al., 1997), and graph-based approaches such as GB-SPM (Wang et al., 2022). Clustering algorithms for trajectories include direct, agglomerative, divisive, hybrid, graph and spectral methods that can be applied using various distance measures (to quantify trajectory similarities) such as dynamic time warping (DTW) and longest common subsequence (LCSS) (Morris and Trivedi, 2009). DTW compare unequal length trajectories by finding a time warping that minimizes the total distance between matching points. LCSS is more robust to noise and outliers than DTW because not all points need to be matched, as instead of a one-to-one matching between points, a point with no good match can be ignored to prevent unfair biasing.

Spatial hotspots are areas or regions where the concentration of geolocated objects within an area is significantly higher than outside that area. Spatiotemporal hotspots are high-density clustering patterns where the number of objects is unexpectedly higher than other observations within a given time interval. Common approaches for hotspot analysis are

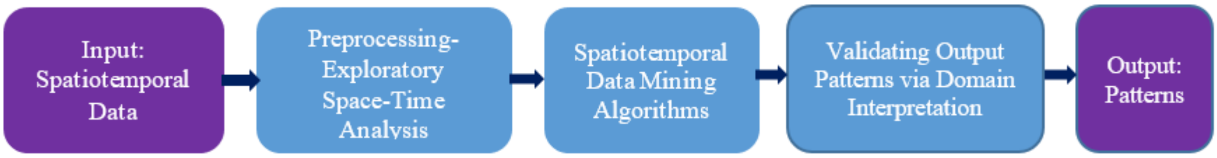

Input: Spatiotemporal Data
Preprocessing-Exploratory Space-Time Analysis
Spatiotemporal Data Mining Algorithms
Validating Output Patterns via Domain Interpretation
Output: Patterns

Figure 2.9.: The process of spatiotemporal data mining (Sharma et al., 2022).

ensemble-based approaches such as Random Forest and Naive Bayes, J48 and decision trees, as well as Fuzzy C-Mean and DBSCAN (Butt et al., 2020). The visualization of hotspots is usually based on estimating the densities of the detected clusters. For example, kernel density estimation (KDE) (Kalinic and Jukka, 2018) or spatiotemporal network kernel density estimation (STNKDE)(Romano and Jiang, 2017) provide visualization of the temporal dynamics of hotspots in the network space and can be used for traffic accident detection.

A *spatiotemporal change* is defined as a change in the statistical distribution of the data, assuming that the data follow a certain distribution. Such an example might be a change in landscape and its temporal dynamics, in order to assess how changes in residential patterns relate to patterns of urban development (Weng, 2007). Depending on the representation of the input data (raster or vector), different approaches are used to detect spatiotemporal changes (Mou et al., 2019; Hong and Vatsavai, 2016).

Although data mining algorithms are an important element in the pattern extraction process, there are additional important elements involved in the chain of the process that transforms raw input data into knowledge (patterns). A general framework of the spatiotemporal data mining process (Sharma et al., 2022) is illustrated in Figure 2.9. The first step of the process is the pre-processing of the often imperfect raw data. Common interventions in this step include data cleaning (cleaning noisy data and errors), data integration when combining data from multiple sources, data selection, data transformation (converting data into a format compatible with the input of data-mining algorithms), and data reduction (e.g., trajectory compression). In this phase, exploratory analysis is often applied using statistical and graphical visualizations (e.g., charts), where anomalies can be discovered and assumptions about the data can be tested. The data are then entered into the data mining algorithm. The output of this step is the discovered patterns, which are explored, evaluated and interpreted by domain experts. If necessary, a refinement process is applied before knowledge is presented (knowledge presentation: the extracted patterns exported in the output).

A data mining framework that explicitly addresses trajectory data is depicted in Figure 2.10 (Alvares et al., 2009). Here the trajectory cleaning process, in addition to noise elimination, may include further checks: i) the computed velocity between two consecutive GPS points must not be greater than a specified threshold, ii) the points in the trajectory must be in time order, iii) each GPS point must have a different timestamp, iv) each trajectory must have more than a minimum number of points (Alvares et al., 2009). Trajectory data, depending on the application and the requirements of the data mining algorithm, are often compressed (Muckell et al., 2014; Yin et al., 2022) for efficient data transmission, storage and retrieval. The key issue in trajectory compression is how to achieve line simplification with low time cost but acceptable loss of accuracy. Finally, matching GPS trajectories (Quddus et al., 2007) with a map (map-matching) often leads to more accurate trajectories

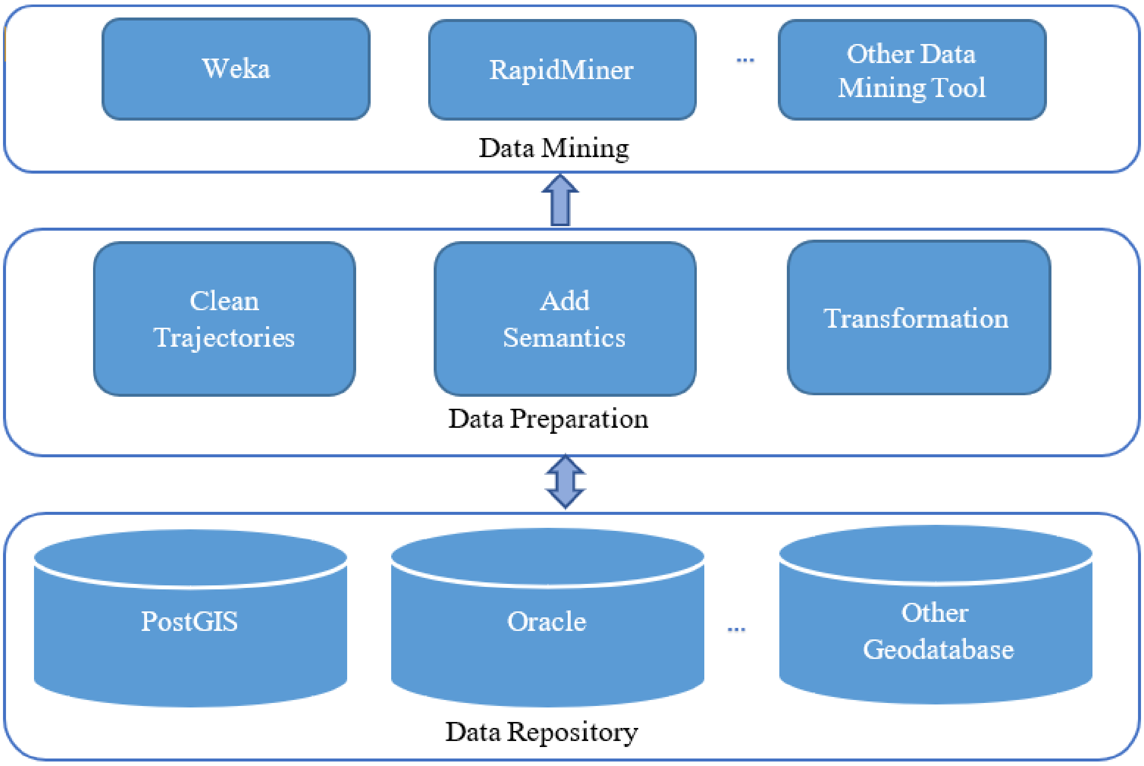


Figure 2.10.: The framework for semantic trajectory mining (Alvares et al., 2009).

and therefore can also be applied in the data preparation stage. For many applications, useful patterns cannot be extracted directly from GPS points and the underlying geographic information must also be taken into account. This information is called *semantics*. The data transformation aims at the same deliverable as the corresponding step in the general data mining framework discussed earlier. After data preparation, the data can be fed into a data mining tool, such as Weka or RapidMiner, where spatiotemporal patterns are detected and extracted to the output. The next Section 2.2.6 explains the concept of semantics in the context of trajectories.

2.2.6. Semantic Enrichment of Trajectories

Raw trajectories can be used by location-based applications when only the location of a moving object is sufficient for the purpose of the application. For example, an application may want to know “where was user A on 24.05.2021 at 07:00 am?”. If the location described by the coordinates 52.383230, 9.693767 is sufficient for what the application wants to calculate, estimate, or infer further, then the trajectories need no further transformation. However, there are applications that aim to richer knowledge about the movement and in addition to the raw trajectories they need additional information related or associated with them. In the previous example, if an application needed to know if user A was at work on 24.05.2021 at 07:00 am, only the coordinates would not be enough to answer the question. In this case, the user’s workplace (Figure 2.11) would somehow have to be geographically defined and annotated as such, so that given the user’s location on that date and time, it could estimate whether or not user A was at her workplace or not. When contextual data is combined with GPS traces, the enriched traces are referred to as *semantic trajectories*. As contextual data can be any data related to the context of the application, e.g., points of

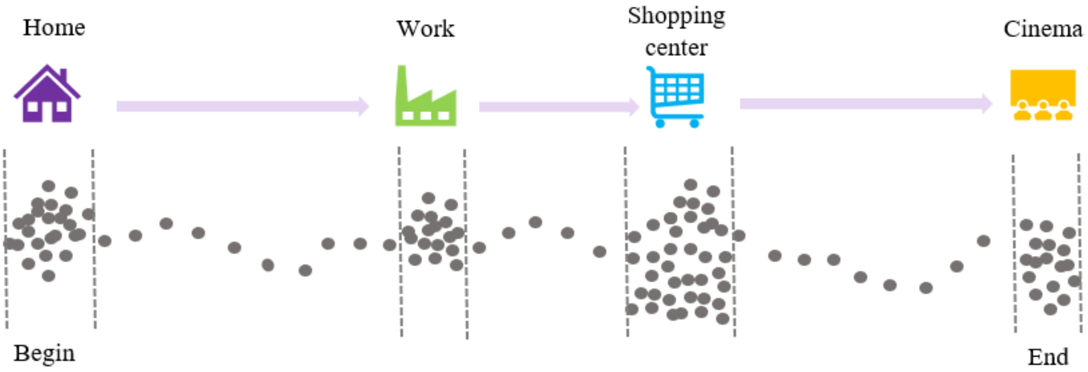


Figure 2.11.: Enriching trajectories with semantic information based on stop and move detections.

interest such as shops, bus stations, museums or street names etc. The process of adding (attaching) knowledge to raw trajectories is known as *semantic enrichment*. In the following, the definition of semantic trajectories is given, as adopted by Parent et al. (2013).

Definition 5. (Semantic trajectory) A *semantic trajectory* is a trajectory that has been enriched with annotations and/or one or several complementary segmentations. It is defined as a tuple:

$(trajectoryID, movingObjectID, trajectoryAnnotations,$
$trace: LISTOFposition\ (instant, point, positionAnnotations),$
$semanticGaps: LISTOF\ gap(t\_gap\_start, t\_gap\_end),$
$segmentations: SETOF\ segmentation\ (segmetationID,$
$episodes: LISTOF\ episode(\ t\_start, t\_end, definingAnnotation, episodeAnnotations)))$
■

Each episode, e.g., a stop at a shopping place, according to the above definition is defined by its start time (t_start) and end time (t_end) and the value of the annotation of the segmentation criterion that applies to this episode ($definingAnnotation$). For example, a criterion to characterize a position as *Home* can be that the moving person remains in that location during the night. Episodes can also have additional annotations ($episodeAnnotations$). A stop episode can include in its annotation list the nearest point of interest found on the map. Episodes may be areas and road intersections through which trajectories pass, means of transport that are used for the movement, category of a region, etc. Because very often episodes in the trajectory of a moving object are associated with the interruption of movement for a period of time, stopping episodes are of particular interest for enriching trajectories with semantics. The aforementioned definition allows the representation of trajectories as sequences of *stops and movements*, where each point of the trajectory belongs to either a stop or a move. Figure 2.11 depicts a GPS track with four semantic locations associated with it, each corresponding to a stop (GPS point clusters). The trajectory in this case can be represented as a sequence of four stops and three moves with the corresponding time period associated with each stopping and moving episode. At a higher semantic level, where each stop represents a specific semantic category (home, work, shop, and cinema) the trajectory can be described as a sequence of stops in semantic locations. With such representations, complex patterns can be identified, such as common sequences of locations visited during weekdays/weekends. In the next Section 2.2.7, we describe the CB-SMoT algorithm that detects stops in trajectories. It also explains why it makes sense to associate stopping episodes with trajectories and intersections crossed by the former, in the context of traffic regulation recognition from GPS data.

2.2.7. Detecting Stop and Moves: the CB-SMoT Algorithm

In light of semantic trajectories and the need to identify complex movement patterns, the research work on extracting stops from trajectories is quite rich. The early work of Ashbrook and Starner (2003) proposes a variant of the *K-means* clustering algorithm, where clusters (*locations*) of points are identified incrementally, starting from a point *place* and examining points within a certain radius of the place point. A location is defined as "any recorded GPS coordinate with a time interval t between it and the previous point". Other simple approaches are based on zero velocity of the moving object, however GPS signal errors can lead to velocity estimates that are often above zero, even though the object is stationary.

Alvares et al. (2007) detect stopping episodes within predefined regions where moving objects remain for at least a minimum amount of time. Zimmermann et al. (2009) address the same problem in error-prone trajectories where some parts of them represent movements that did not actually occur (e.g., the vehicle was stopped but due to GPS error or device inefficiency, that part of the trajectory is represented as moving) using an interactive density-based clustering algorithm, where the density is defined by spatial and temporal criteria and the user can adjust the parameters of the algorithm in an interactive way using a visualization tool. Rocha et al. (2010) consider the problem of identifying interesting locations in ocean fishing vessel trajectories with a rather low sampling rate (one sample per thirty minutes), taking into account the change of direction as the main aspect. Finally, the clustering method proposed by Xiang et al. (2016) considers the spatial and temporal proximity between successive points (point sequence) within trajectories, based not on the speed of each point, but on the speed of the sequence.

The latter algorithm is methodologically very close to the Clustering Based Stops and Moves of Trajectories (CB-SMoT) detection algorithm (Palma et al., 2008). CB-SMoT is presented in detail below, as this thesis proposes a modification of it (Section 4.3.1) for detecting short-duration stopping and decelerating events on individual trajectories, which are then used as classification features in the context of the traffic regulation recognition problem. CB-SMoT can be seen as a variant of the well-known density-based DBSCAN (Ester et al., 1996) clustering algorithm, where not only the spatial distance between points but also their temporal proximity is taken into account. More specifically, CB-SMoT modifies the following aspects of the DBSCAN algorithm: (i) instead of searching for a minimum number of points within the neighbourhood, points are clustered based on a minimum temporal distance criterion; and (ii) the definition of a point's neighbourhood is adapted to take into account further temporal constraints between points (cf. Definition 9). In the following, we present all definitions involved in CB-SMoT, as given in the original paper (Palma et al., 2008), as well as the algorithm itself.

Stops represent important positions within the trajectory where the moving object remains for at least a minimum amount of time. Important locations, as areas with a defined geometry, are defined taking into account the context of the application used and correspond to different types of spatial features defined in a geographical database. For each relevant spatial feature type, a minimum time interval is defined so that a trajectory must continuously intersect the area of that feature type to be considered a stop. A particular spatial area that is crossed by a minimum time from a trajectory, is called a candidate stop.

Definition 6. (Candidate Stop) A *candidate stop* C is a tuple (R_c, Δ_c), where R_c is a topologically closed polygon in $\mathbb{R}^2$ and Δ_c is a strictly positive real number. The set R_c is called geometry of the candidate stop and Δ_c is called its minimum duration. The candidate stops are dependent of specific applications. A (predefined) location-based application A is a finite set

$$A : \{C_1 = (R_{c_1}, \Delta_{c_1}), C_2 = (R_{c_2}, \Delta_{c_2}), \ldots , C_N = (R_{c_N}, \Delta_{c_N})\}$$

of candidate stops with non-overlapping geometries R_{c_1}, R_{c_2}, ..., R_{c_N}. ■

Definition 7. (Stop) A *stop* episode[8] of a trajectory T with respect to an application A is a tuple (R_k, t_j, t_{j+n}), such that a maximal subtrajectory of T:

$$\{(x_i, y_i, t_i) \mid (x_i, y_i) \; intersects \; R_k\} = \{(x_j, y_j, t_j), (x_{j+1}, y_{j+1}, t_{j+1}), ..., (x_{j+n}, y_{j+n}, t_{j+n}\},$$

where R_k is the geometry of $C_k \in A$ and $|t_{j+n} - t_j| \geq \Delta_k$, $k \in \{c_1, c_2, ..., c_N\}$ and $n > j$. ■

Definition 8. (Move) A *move* of a trajectory T with respect to an application A is:

i a maximal contiguous subtrajectory of T in between two temporally consecutive stops of T; or

ii a maximal contiguous subtrajectory of T in between the starting point of T and the first stop of T; or

iii a maximal contiguous subtrajectory of T in between the last stop of T and the last point of T; or

iv the trajectory T itself, if T has no stops.

■

Therefore, any point on a trajectory that is not part of a stopping episode is a movement. A motion has no minimum duration, and may or may not intersect a candidate stop. If it does, the intersection interval must be less than the minimum duration of the stop candidate. It should be noted that the minimum stop duration shall be determined taking into account the sampling period of the trajectory points to ensure that there are enough points to characterise a stop.

Definition 9. (Eps-linear-neighborhood) Let $\{p_0, p_1, ..., p_k, p_{k+1}, \ldots ,p_N\}$ be a trajectory, where $p = (x, y, t)$ represents a point of the sequence of the points of the trajectory. The *Eps* linear neighborhood of a point p_k, denoted by $LNEps(p_k)$, is the maximal set of points p_i, such that:

$$(\sum_{i=m}^{k-1} dist(p_i,p_{i+1})) \leq Eps \cup (\sum_{i=k+1}^{n} dist(p_{i-1}, p_i))) \leq Eps, \; with \; t_0 \leq t_m < t_k < t_n \leq t_N$$

---

[8] The term *episode* in the definition is an insertion by the author of the thesis and has been adopted to differentiate stops referring to episodes within a single trajectory from significant locations which are also called stops and refer to places of global interest, i.e., places where many trajectories stop.

Eps is a positive number representing the maximum distance between a point p and its neighbours in the trajectory. Instead of considering a minimum number of points to define a region as dense (cluster), the minimum dwell (stay) time in a region is used. ■

Definition 10. (Core point) A point $p = (x_p,\ y_p,\ t_p)$ of a trajectory is called *core point* with respect to Eps and $MinTime$ if $|t_n - t_m| \geq MinTime$, where n is the last point of $LNEps(p)$, and m is the first one (the neighborhood is a time-ordered point sequence). ■

An important observation here is that the core point implies a maximum velocity constraint. The ratio $\frac{Eps}{MinTime}$ gives the maximum average velocity of the corresponding linear neighborhood. Increasing of the parameter $MinTime$, the relative velocity decreases. Besides, using time constraint instead of number of points eliminates the influence of problems related to the limited number of points due to device or signal failure during sampling or the sampling rate itself.

Definition 11. (Directly density-reachable) A point q is *directly density-reachable* to a point p, if $q \in LNEps(p)$ and p is a core point with respect to Eps and $MinTime$. ■

Definition 12. (Density-reachable) A point q_0 is *density-reachable* from a point p with respect to Eps and $MinTime$, if there exists a chain q_0, q_1 q_2, ... q_N where $q_N = p$ and q_k is directly density-reachable to q_{k+1}. ■

Definition 13. (Density-connected) Two points p and q are *density-connected* with respect to Eps and $MinTime$, if there exists a point o and both p and q are density-reachable from o. A non-core point can be density-connected to another non-core point if both have a common core point. ■

Definition 14. (Trajectory cluster) A *trajectory cluster* G of a trajectory T with respect to Eps and $MinTime$ is a non-empty subtrajectory of T formed by a set of contiguous time-space points such that:

1. $\forall p,\ q \in\ T$: if $p \in G$ and q is density-reachable from p with respect to Eps and $MinTime$, then $q \in G$.
2. $\forall p,\ q \in\ G$: p is density-connected to q with respect to Eps and $MinTime$.

■

The CB-SMoT algorithm (Algorithm 1) is a two-step algorithm that takes as input a single trajectory and two parameters, *Eps* (see Definition 9) and $MinTime$ (see Definition 10). In the first step, stop episodes are identified on the trajectory (Figure 2.12). In the second step, the algorithm determines where these stop episodes (clusters) are located, taking into account the application context A of the trajectory as defined by the list of candidate stops (see Definition 6). The algorithm at this stage checks if the cluster intersects for at least $MinTime$ one of the candidate stops. If not, the detected stop episode is considered as an interesting place and is added to the list of unknown stops. An example of detected unknown stop locations as well as of stop episodes detected in known stop locations (candidate stops) is illustrated in Figure 2.13. Each unknown stop receives an identifier, and if two or more unknown stops intersect, as in Figure 2.12, they receive the same identifier.

Definition 15. (Unknown Stop) An *unknown stop* of a trajectory T with respect to an application A, *Eps*, and $MinTime$, is a cluster G_k of T which does not intersect an R_j of A for at least Δ_j , where $C_j = (R_j,\ \Delta_j)$ is a candidate stop. ■

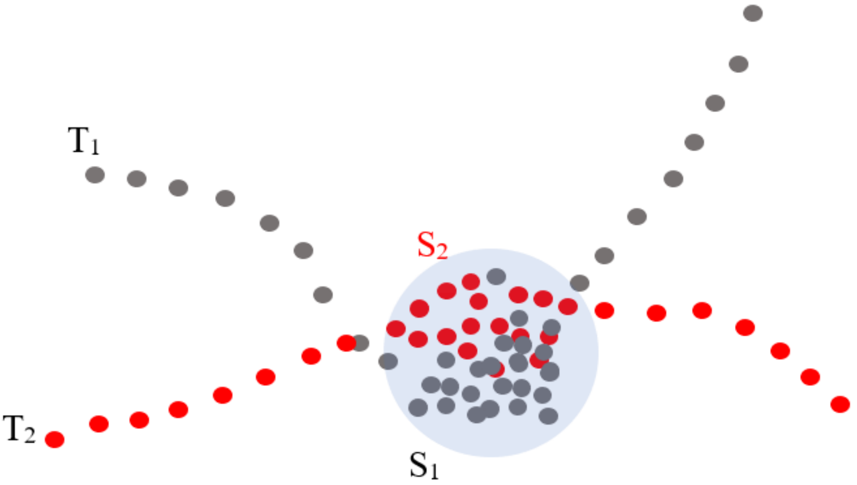


Figure 2.12.: Two trajectories T_1 and T_2 with stop episodes S_1 and S_2 on the same location (blue).

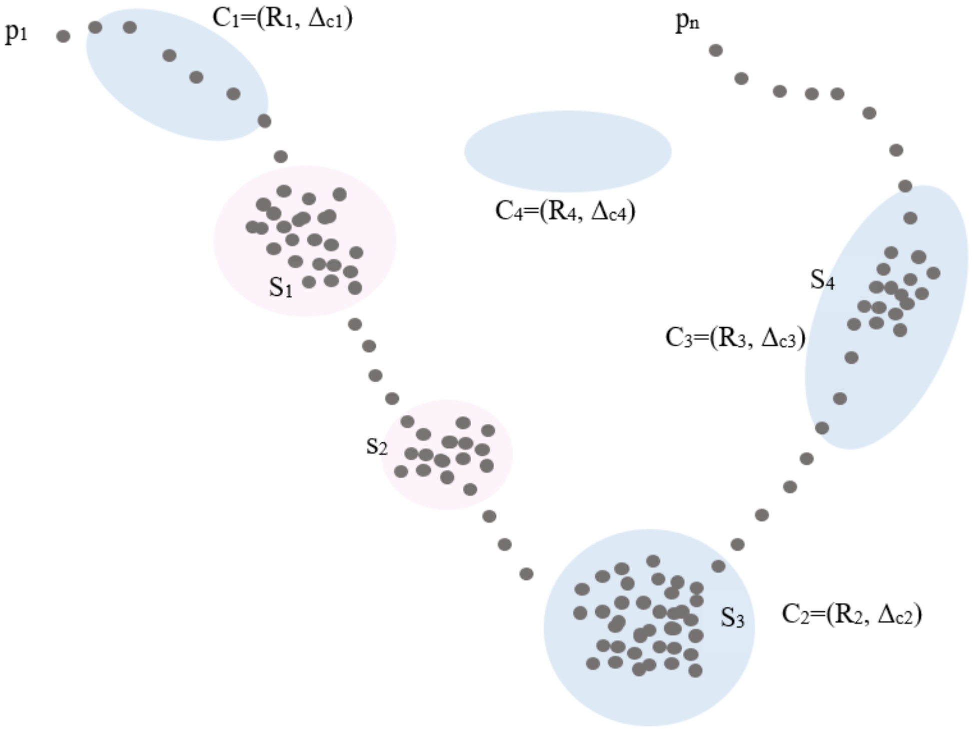


Figure 2.13.: A single trajectory with starting point p_1 and ending point p_n. The region which the trajectory is observed on, includes four candidate stops C_1, C_2, C_3, C_4 (marked in blue), at two of which the moving object stops (S_3 stops at C_2 and S_4 at C_3). CB-SMoT can also detect unknown stopping locations, such as S_1 and S_2 (marked in pink). Although C_1 is crossed by the trajectory, no stopping episode is detected at it, as the requirement for staying more that Δ_{c_1} is not met.

Algorithm 1: The CB-SMoT algorithm: clustering-based stop and move detection of trajectories.

```
Data:
T : a GPS trajectory
A : application
Eps (param) : linear point neighborhood distance
MinTime(param) : minimum time for clustering
Result: CB-SMoT identifies clusters of consecutive points within a trajectory that
        remain at least MinTime within a linear neighborhood, defined by the
        parameter Eps.
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
// Step 1: stop (clusters) detection
foreach unprocessed point p of T do
    // find the neighbors of p
    neighbor_list = linear_neighborhood(p, Eps)
    if p is a core point wrt Eps, minTime then
        for each neighbor n in neighbor_list do
            N_neighbor_list = linear_neighborhood(n, Eps)
            neighbor_list = neighbor_list ∪ N_neighbor_list
        end for
        add neighbor_list as cluster with cluster_id in Clusters
        set all points in neighbor_list as processed
    end if
end foreach
// Step 2: identify if a stop episode intersect a known location (candidate stop) or
// recognise it as a unknown stop location
foreach cluster in Clusters and each C : ($R_c$,$\Delta_C$) in A do
    if cluster intersects $R_c$ for time interval $t \geq \Delta_C$ then
        generate a stop in Stop_list
    end if
    else
        generate a unknown stop in Stop_list
    end if
end foreach
for each subtrajectory which is not a stop (not in clusters) do
    generate a move in Move_list
end for
return Stop_list, Move_list
```

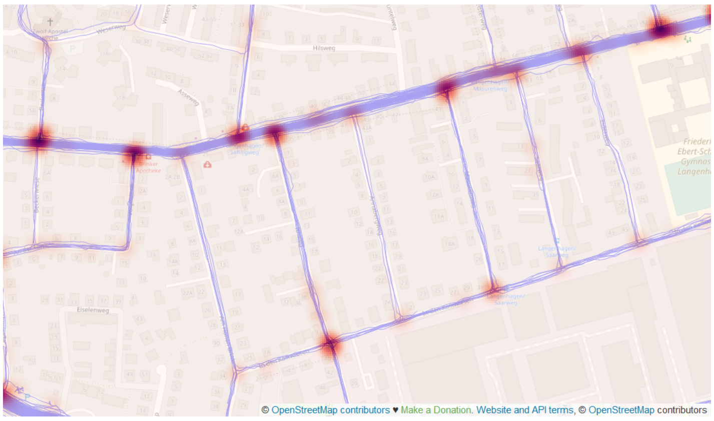
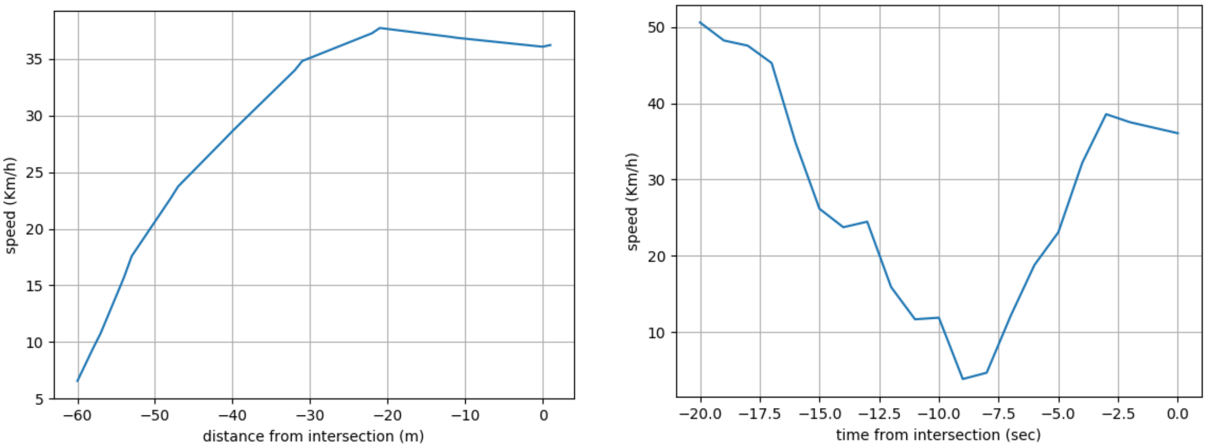

© OpenStreetMap contributors ♥ Make a Donation. Website and API terms, © OpenStreetMap contributors

Figure 2.14.: Hotspots where the speed of vehicles is less than 10 kmh (red color). In blue are depicted the trajectory samples.

For the scope of this thesis as explained later in Paragraph 4.3.1, the CB-SMoT algorithm will be modified for detecting stop and deceleration episodes in single trajectories, being motivated by the following observation. Figure 2.14 shows the hotspots, detected using the open source geographic information system QGIS, where the speed of vehicles is less than 10 kmh. It is clear that the locations where such low-speed incidents are observed are near intersections and this further motivates the idea of using such detections for the recognition of the traffic controls of intersections. In addition, Figure 2.15 depicts the speed profiles of a vehicle approaching an intersection regulated by a traffic signal. From 60 m to 21 m measured from the centre of the intersection (left graph), the speed increases continuously from 5 kmh to 38 kmh and remains just above 35 kmh until it crosses the centre of the intersection. The right graph shows the trajectory speed of the same vehicle in the last 20 s before crossing the intersection (the center of the intersection is at 0 s). Here we see that the vehicle decreases its speed while approaching the intersection until it almost stops 9 s before the center of the intersection. It then begins to accelerate similar to the left diagram.



(a) (b)

Figure 2.15.: Speed profile of a vehicle before crossing a traffic light controlled intersection. Both graphs correspond to the same vehicle trajectory. (a) shows the speed of the last 60 meters, while (b) shows the speed of the last 20 seconds, both measured from the centre of the intersection.

From these graphs (speed profiles) it is clear that low speed episodes, which can be either stopping or deceleration episodes can characterize locations such as intersections, and therefore can be useful for the scope of detecting their control type (regulation category). Section 2.3 presents the machine learning algorithms used to implement the proposed methodologies of this thesis.

2.3. Machine Learning

In this section, we explain some basic concepts of machine learning. First, the learning types of machine learning are clarified in Subsection 2.3.1. Subsection 2.3.2 deals with supervised learning methods and describes two classification methods, Random Forest and Gradient Boost. Subsection 2.3.3 is concerned with unsupervised learning and describes two well-known clustering techniques, K-means and DBSCAN. Subsection 2.3.4 discusses two semi-supervised methods, self-training and cluster-then-label, that address the problem of label sparsity. Subsection 2.3.5 explains the concept of active-learning, which focuses on how to select training data so that better learning, in terms of classification performance, is achieved with less training data. Subsection 2.3.6 explains how learning can be implemented in an incremental manner, where learning models process one observation (or a mini-batch) at a time, as opposed to batch learning, where models are learned by processing all the data at once.

2.3.1. Machine Learning and Types of Learning

The field of machine learning is concerned with the question of how computer programs can learn a learning task and automatically improve with experience. Mitchell (1997) defines the learning broadly, to include any computer program that improves its performance at some task through experience. More precisely:

Definition 16. (Learning) A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. ■

For example, a computer program that learns to play chess might improve its performance, as measured by its ability to win in the set of tasks that involve playing chess games, through the experience gained by playing games against itself or other opponents.

The process of solving a practical problem with machine learning generally involves five steps (Zheng and Casari, 2018, p. 4), which are illustrated in Figure 2.16. First, the goal(s) must be clarified, in terms of what a computer program must learn to do, given a set of input data. In the second step, the data required for the learning task is collected and then processed to have a format compatible with that which the computer program accepts as input. In the third step, called feature engineering, the raw data is converted into features (variables). Features represent measurable pieces of data, and finding the right features is crucial for the learning capability that the computer program will obtain in the next steps (e.g., how accurate predictions it can make). Creating features from raw data or identifying them from raw data often requires some domain knowledge expertise for the problem of study. During feature engineering, data that are not useful for the particular learning task are eliminated, missing values are predicted, data from different sources are integrated, and outliers or anomalies in the data are excluded from the input dataset and ultimately are not desirable to be learned by the algorithms. Clearly, this whole process requires some kind of data analysis, which can identify the aforementioned peculiarities that a dataset may have and then motivate appropriate actions for addressing them. For this reason, feature engineering additionally involves an exploratory data analysis that sheds light on the content and quality of the data from which the machine learning algorithms will learn the specific learning task. Often such an analysis reveals unique features, identifies those that are less intuitive, and leads to hypotheses about the object of study. Once the features are determined and the data is cleaned, the dataset is split in training, validation and test datasets. A learning model is constructed from the training data, in a process called *training*. The model both sees and learns from this data. For tuning the model hyperparameters, the validation dataset is used. The model occasionally sees this data, but never *learns* from this. Learning is done only with training data. Then, the test data (unseen data), which has not been used so far in any of the previous steps, are used to evaluate the learning ability of the model acquired through the training process, using some performance metrics. There are four types of learning: *supervised*, *semi-supervised*, *unsupervised* and *reinforcement*.

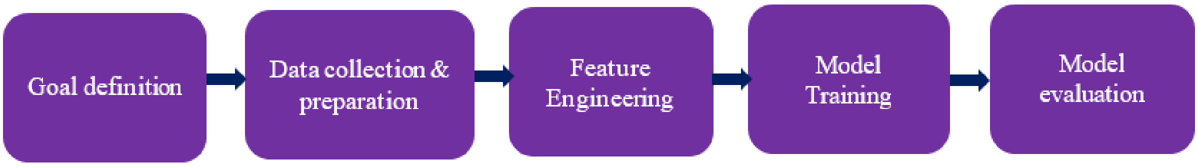


Figure 2.16.: Implementation steps for solving a problem with machine learning.

Supervised learning uses labeled data to train algorithms that can either classify (label) data into categories (classification) or predict outcomes (regression). In particular, the goal of the learning task is to learn a function that maps an input to an output based on pairs of data (also called examples) that have the form of input and output. These input-output pairs are denoted as $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, where N is the number of examples and the dimension of the feature vector $\mathbf{x}$ is equal to the number of features, i.e., $\mathbf{x} = [x^{(1)}, x^{(2)},..., x^{(D)}]$, assuming that the number of features is D. The label y_i can be either a categorical value belonging to a finite set of classes 1, 2, ..., C, or a real number, or a more complex structure, such as a matrix or a graph. In the context of this thesis the output will always be a class (corresponding to a traffic regulator).

In unsupervised learning, the dataset is a collection of unlabeled examples, denoted as $\{\mathbf{x}_i\}_{i=1}^{N}$, with $\mathbf{x}$ is the feature vector. The goal of unsupervised learning is to build a model by taking a feature vector $\mathbf{x}$ as input and returning either a value that has a particular meaning in the context of the problem of study, or another vector that again is itself of particular meaning. For example, in clustering, which involves grouping a particular set of objects or entities based on their characteristics and aggregating them according to their similarities, for each input $\mathbf{x}_j$, the learning model returns the identifier of the cluster to which it belongs. For example, suppose that the data are clustered into m clusters, identified as 1, 2,..., m, then for each input data, the clustering algorithm returns a number $k \in \mathbb{Z}$ and $k \in [1, m]$. In dimensionality reduction, the output of the learning model is a feature vector $\mathbf{x} = [x^{(1)}, x^{(2)},..., x^{(D)}]$ that has fewer features than the input $\mathbf{x} = [x^{(1)}, x^{(2)},..., x^{(n)}]$ $(D < n)$.

In semi-supervised learning, the dataset consists of both labeled and unlabeled examples, with the set of unlabeled examples being much larger than the set of labeled examples. It is often difficult to find a large dataset of labeled examples, and labeling is challenging because it practically involves a manual process that is both time consuming and costly. Semi-supervised learning combines supervised and unsupervised learning and aims to create better learners by using labeled and unlabeled data than by using each one individually. Semi-supervised problems include semi-supervised classification, regression, and clustering.

Reinforcement learning is an approach to control learning that accommodates indirect or delayed feedback as training information (Mitchell, 1997). An autonomous agent (a machine) that lives, perceives, and acts in an environment can learn to choose optimal actions to achieve its goals through a training process that provides rewards and punishments (penalties) for the actions performed according to their desirability (rewards for desirable actions and punishments for incorrect actions). The goals in reinforcement learning can be very general and range from learning to play a board game to controlling a robot. In the next paragraphs we explain the supervised, unsupervised and semi-supervised methods used to implement this thesis.

2.3.2. Supervised-Learning: Classification

As mentioned earlier, the problem addressed in this thesis is a classification problem, and hence in this section we will focus on it. Classification is the task of assigning labels to unlabeled instances of data, and this task is performed by a classifier. A classifier is usually described in terms of a *model*. The model is created using a given set of instances (examples), known as the training set, which contains feature values as well as class labels for each instance. In addition to the training set, learning algorithms are also required in order to learn a classification model in a systematic way. The process of using a learning algorithm to create or build a classification model from the training data is known as *induction*. This process is also often described as *learning a model* or *building a model*. The process of applying a classification model to unseen test instances to predict their class labels is known as *deduction*. Thus, the classification process involves two steps: 1. a learning algorithm is applied to the training data to learn a classification model, and 2. the classification model is applied to unlabeled instances to assign (predict) labels (Tan et al., 2018, 137).

There are many different types of classifiers. Tan et al. (2018) categorize them according to their output characteristics, into binary versus multi-class and deterministic versus probabilistic. Furthermore, depending on the technique used to distinguish instances from different classes, into linear vs. nonlinear, global vs. local, and generative vs. discriminative classifiers.

In short, binary classifiers assign each instance to one of two possible labels, usually denoted as +1 and -1. If classifiers must assign instances to more than two labels, then the classifier is called a multiclass or multi-category classifier. A deterministic classifier assigns each instance of data a discrete-value label, while a probabilistic classifier assigns a continuous value between 0 and 1 indicating how likely it is that an instance belongs to a particular class, with the probability scores for all classes summing to 1. Some examples of probabilistic classifiers include the Naive Bayes classifier, Bayesian networks and logistic regression.

A linear classifier uses a linear separating hyperplane to distinguish instances from different classes, while a non-linear classifier allows the construction of more complex, non-linear decision boundaries. The assumption that classes can be distinguished by a linear hyperplane leads to simple models that on the one hand are less prone to overfitting, on the other hand are not flexible enough for capturing complex data. An example of a linear classifier is the perceptron and a non-linear one is the multi-layer neural network. A global classifier fits a single model to the entire dataset, while a local classifier divides the input space into smaller regions and fits a different model to the instances of each subregion. Depending on whether or not classifiers need to know the underlying mechanism that generates the instances of all classes (distributions), they are divided into generative and discriminative. Discriminative models make predictions on unseen data instances without explicitly describing the distribution of each class label, whereas a generative model needs to know the distribution of the dataset (classifiers learn a generative model for each class) in order to return a probability for a given instance. Some examples of discriminative classifiers include decision trees, nearest neighbor classifier, artificial neural networks, and support vector machines, while examples of generative classifiers include the Naive Bayes classifier and Bayesian networks. The following section focuses on two popular classification techniques, decision trees and ensembles of decision trees, which according to the aforementioned categorization fall into the categories of multiclass, non-deterministic, non-linear and discriminative.

2.3.2.1. Decision Trees

Decision trees are commonly used models for classification and regression tasks. Here they are discussed in the context of classification. The basic idea behind decision trees is that they lead to certain decisions by learning a hierarchy of if-else conditions (questions) from the data. For example, suppose someone wants to decide whether to go to work or not, given that he goes to work unless it is a weekend and no urgent work is pending, or if it is a weekday and a holiday and no urgent work is pending. Clearly, the circumstances that determine whether he goes to work are whether it is a weekend, whether there is urgent work pending, and whether it is a holiday. These conditions take binary values (true, false) and correspond to the attributes of the classification tree. Thus, the decision can be made by asking simple questions such as whether it is a weekend, etc. Such a set of questions can be expressed in the form of a tree, as shown in Figure 2.17. Each node of the tree represents either a question or a terminal node (also known as a leaf) containing the answer (decision). The top node is called the root. Therefore, to model such a decision tree with machine learning, three binary attributes (is_weekend, pending_urgent_task and is_holiday) would be needed and the output would be binary and would correspond to true if the decision is to go to work, or false to stay at home. Then, given a dataset and the associated learning algorithm, the decision tree is learned from the data.

The problems typically addressed by machine learning are more complex than the example above. Also, the questions (features) that lead to decisions usually have continuous values as answers are rather than just yes-no. To present a realistic example we will use the well-known Iris flower dataset, which initially consists of 4 instances of features (sepal length, sepal width, petal length, petal width) with a target that can receive three possible labels (iris Setosa, iris Versicolour, iris Virginica). For visualization purposes (in 2D), we eliminate the number of features to two (sepal length and sepal width) and the number of classes also to two (Setosa, Versicolour). Since the features have continuous values, the questions are of the form “whether the value x of the feature is greater than the value v?”. Then, the learning algorithm at each step tries to find from all possible questions[9] the one that is most informative (according to some criteria) for the target variable. The first question, represented as a root node, represents the entire data set. Each question leads to a split and is either followed by another question or the recursive process of questions results in a leaf when all data corresponding to the question have the same label (such a leaf is called pure). Given a tree constructed from a training dataset, predictions can be made on a new data instance by running the nodes from top to bottom and following each time the part of the tree that is true for the particular feature of the data instance.

The problem with creating decision trees in the way described above is that they can become arbitrarily deep and memorize training examples, as shown in Figure 2.18e and 2.18f. Tree depth is defined as the length of the longest path from a root to a leaf. In this example the tree has a depth of four and all leaves are pure (the accuracy is therefore 100%), and we can see from Figure 2.18e that all instances are correctly classified. There are several strategies for preventing overfitting and motivating better generalization to unseen data. Some common techniques include limiting the depth of the tree, the number of terminal nodes, and the minimum number of instances required to split an internal node. Stopping the tree early before it is fully grown, using a strategy such as those mentioned earlier, is known as pre-prunning. Such examples are shown in Figures 2.18b and 2.18d, where the maximum depth of the tree is set to one and two, respectively. The first tree is quite simple and its discriminative ability to distinguish examples from the two classes is quite poor, as shown in Figure 2.18a, where there are many misclassified instances. In contrast, the depth-two tree, although still simple, manages to separate the two classes very well and without using elaborate hyperplane (unlike the example in 2.18e). Another possibility is to let the tree fully grow and then remove nodes that contain little information. This strategy is called post-pruning.

---

[9] Each question is called *attribute test condition.*

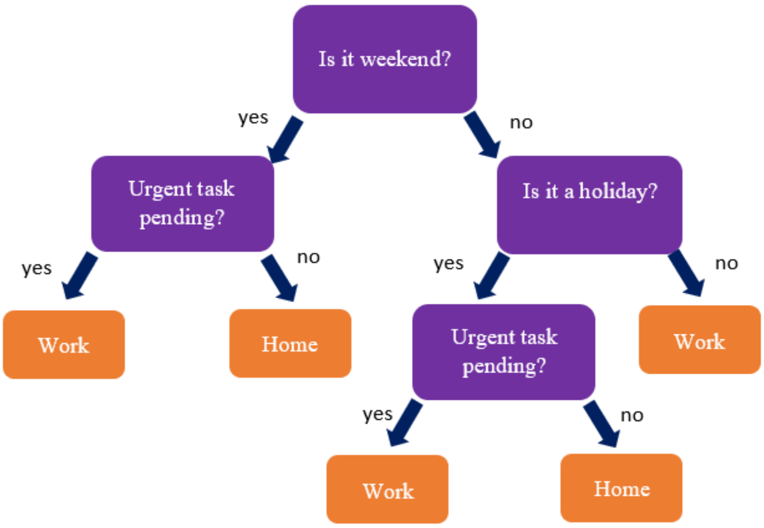


Figure 2.17.: A decision tree that decides whether someone goes to work or stays at home. The binary features (true, false) are is_weekend , pending_urgent_task, is_holiday. Someone goes to work unless it is a weekend and there isn't any pending urgent task at work, or it is a weekday and a holiday and no urgent work is pending at work.

A measure for a node's impurity is the gini score: when a node is pure, all training instances it applies to belong to the same class and gini score is equal to zero. For example, all the leaf nodes of the depth-four tree (Figure 2.18f) have gini score equal to zero. Equation 2.1 shows how the training algorithm computes the gini score G_n of the nth node. The depth-two (Figure 2.18b) left node has a gini score equal to $1 - (0/51)^2 - (45/51)^2 - (6/51)^2 \approx 0.207$.

$$G_n = 1 - \sum_{c=1}^{M} p_{n,c} \tag{2.1}$$

where $p_{n,c}$ is the ratio of class c instances among the training instances in the n_{th} node. An alternative impurity-based measure to gini score is the entropy (Lee et al., 2022) .

A known algorithm to train binary decision trees is the Classification And Regression Tree (CART) algorithm (the Python library Scikit-Learn uses it). CART first splits the training set in two subsets using a single feature f and a threshold $thres_f$ (e.g., sepal length <= 5.45 cm). f and $thres_f$ values are found by searching for the pair $(f,thres_f)$ that produces the purest subsets weighted by their size. The cost function that CART tries to minimize is given by Equation 2.2. Once it has successfully split the training set in two subsets, it splits the subsets using the same process and continues the this recursive process until either the tree has grown at the maximum defined depth or it cannot find a split that will reduce further its impurity. Because CART splits nodes always to two parts, it can only produce binary trees. Other algorithms can create trees with nodes that have more than two children, as for example the ID3 algorithm.

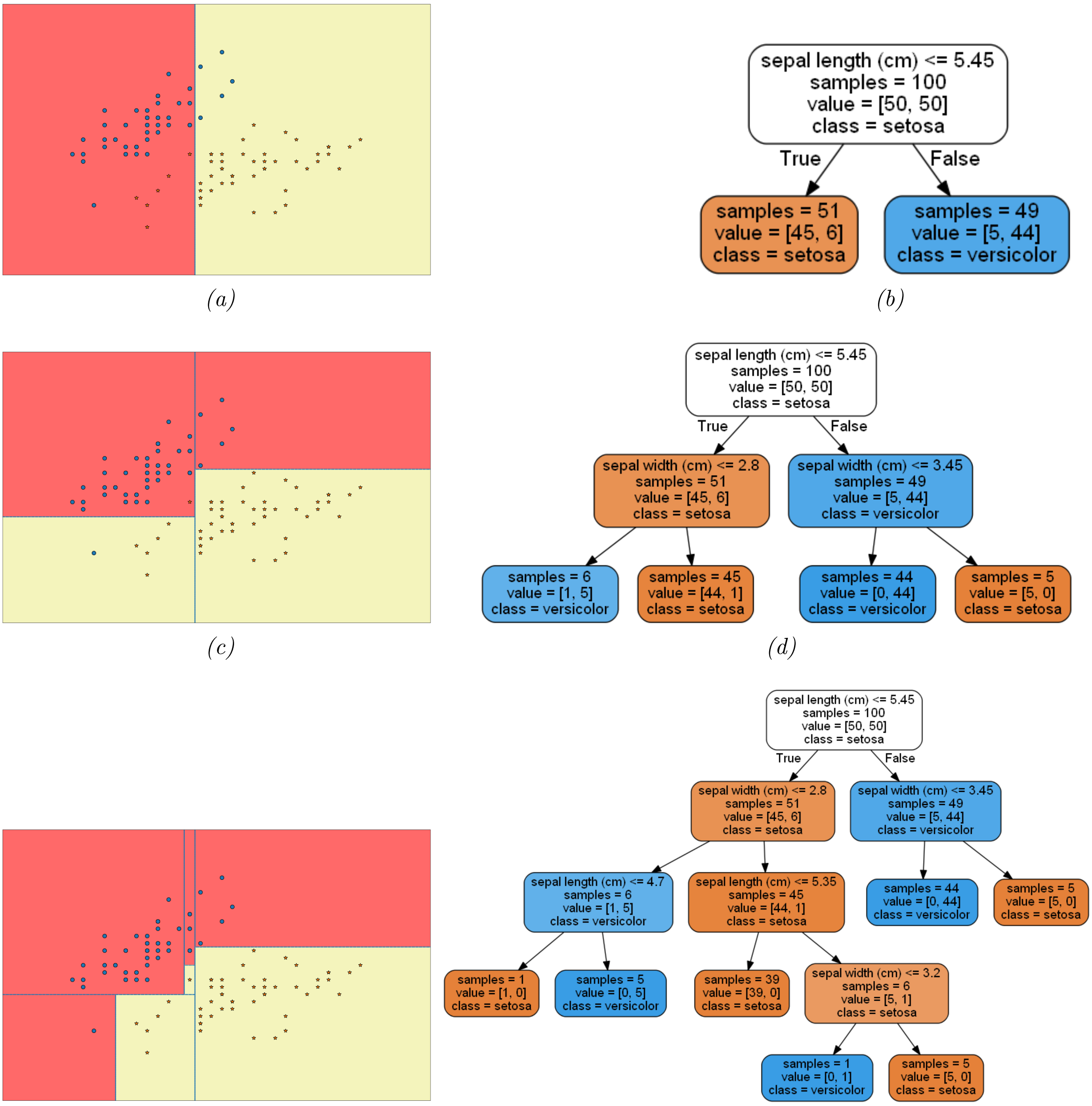
$$L(f,thres_f) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right} \tag{2.2}$$

where $G_{left/right}$ is the impurity score of the left/right subset, $m_{left/right}$ is the number of instances in the left/right subset, and m the total number of instances of the node to be partitioned.

The main advantage of decision trees is that they are simple to understand and interpret (they can be visualized), even by non-experts, they are flexible and the algorithms are scale invariant. The latter comes from the fact that each feature is processed separately and therefore the splitting process does not depend on the scale of the features. This means that features do not need to undergo any feature scaling, such as normalization or standardization in a preprocessing stage. On the other hand, their main limitation is that they tend to overfit and therefore provide poor generalization. Decision trees use orthogonal decision boundaries, which makes them sensitive to small changes in the training data (e.g., rotation). One way to address this problem is to use several decision trees (ensemble of trees) instead of just one. This issue is addressed in the next section.

2.3.2.2. Ensembles of Decision Trees

Ensembles are methods that combine the predictions of multiple learning models to create more powerful models. For example, instead of relying only on the prediction of a Logistic Regression classifier on a data instance, the prediction could be combined with the predicted label of another classifier, for example an SVM classifier (Géron, 2019, p. 189). This can be done by aggregating the predictions of different classifiers, generally using a "voting" strategy. Two main voting strategies are *hard* and *soft* voting, which are illustrated in



(a) *(b)*

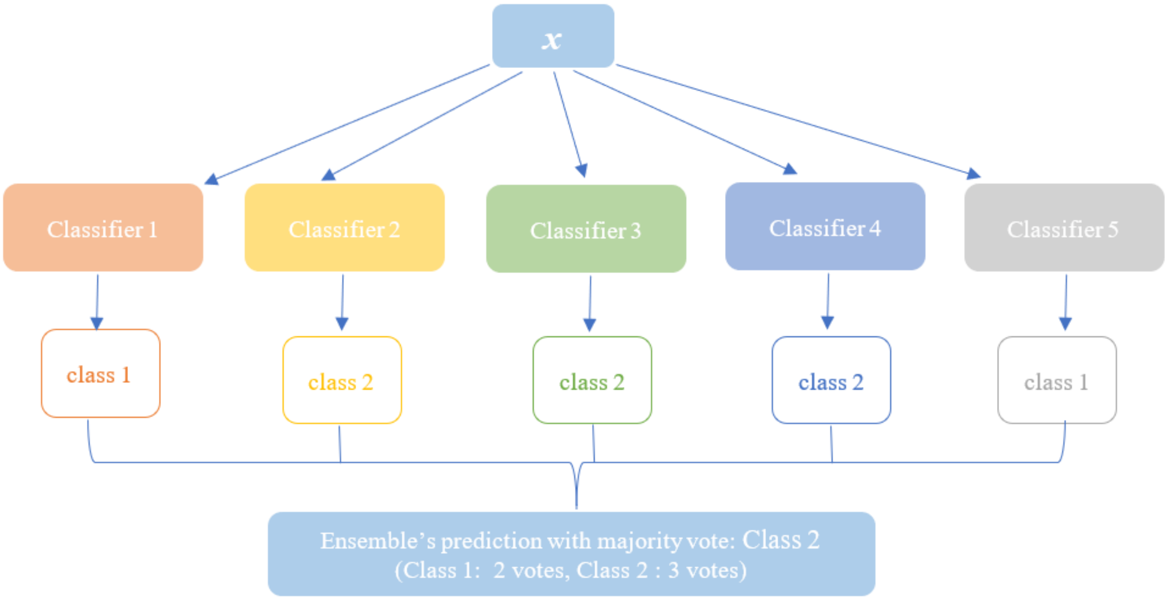
(c) *(d)*

(e) *(f)*

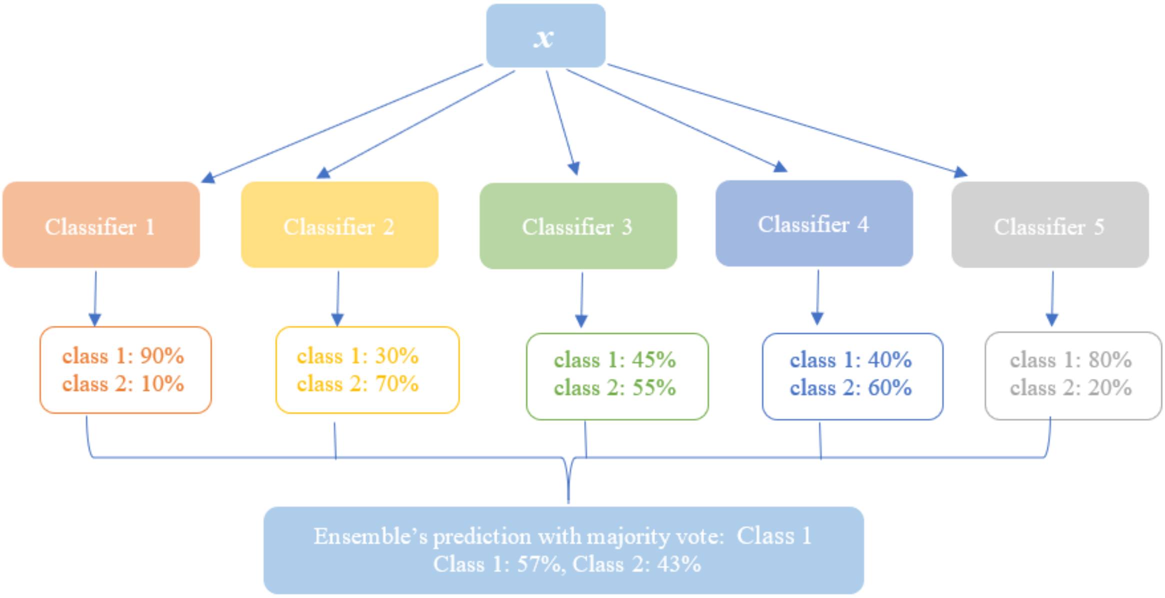
Figure 2.18.: An example of a binary decision tree on the modified Iris dataset (2 classes) for different (max) values of depth: (a-b) 1, (c-d) 2 and (e-f) 10.

Figure 2.19. Hard voting, e.g., majority vote, aggregate cumulatively the predicted labels of the classifiers and assigns as label the one predicted by most classifiers (Figure 2.19a). When the classifiers can estimate the probabilities of the classes, the soft voting strategy averages the probabilities of all classifiers per class and predicts the class with the highest average value (Figure 2.19b).

There are many ensemble models in the machine learning literature. Two very effective ones that use decision trees as ensemble learning models are *random forests* and *gradient boosted* decision trees, which are discussed in the next section. In both cases, instead of using different learning algorithms (different classifiers) as in the example with Logistic Regression and SVM mentioned earlier, the ensemble uses the same training algorithm (decision tree) for each predictor, but each predictor is trained differently, as discussed in the next section.



(a) Hard voting involving five classifiers.



(b) Soft voting involving five classifiers.

Figure 2.19.: Voting classifiers. Class 1: (90% + 30% + 45% + 40% + 80%)/5 = 57%. Class 2: (10% + 70% + 55% + 60% + 20%)/5 = 43%.

Random Forest A random forest is an ensemble (group) of decision trees, where all trees predict the same target (variable) and each tree is constructed in such a way that it differs slightly from the other trees in the ensemble. To create trees that are similar but neither the same nor very different, randomness is involved in the decision tree learning process and for this reason the set of randomized decision trees is called a random forest. The motivation for creating random trees comes from the fact that decision trees, although good predictors, often overfit, so by creating many slightly different trees and averaging their predictions then the overfitting can be eliminated, assuming that each tree overfits in a different way from the other trees in the ensemble. Therefore, the challenge in a random forest is to construct many good predictors of the same variable, all decision trees, each different from the other.

There are two ways in which randomized trees are created: by selecting (sampling) the data instances to create a tree and by selecting the features to be used to partition the tree. There are two strategies for selecting data instances: *bagging* and *pasting*, which in combination with feature selection can lead to *random patches* and *random subspaces*.

The *bagging sampling* samples data instances from the initial dataset by replacement. The resulting dataset is called a *bootstrap*. This process creates a dataset that is different from the original dataset, as some of the instances will be the same as the original dataset, but some will be missing. The *pasting sampling* samples data instances without replacement. In other words, both bagging and pasting allow sampling training instances several times across all the different predictors in the ensemble, but only bagging allows sampling training instances multiple times for the same predictor.

In the tree building process, features can also be sampled, similar to data instances. In this way, each predictor can be trained on a random subset of the input features. The sampling of both the training instances and the features is called *random patches* sampling. Retaining all training instances of the original dataset and sampling only the features is called *random subspaces* sampling. Therefore, unlike decision trees where for each node the best feature is searched for splitting the dataset, in the random forest for each node the algorithm searches for the best split in a random *subset* of features. This feature sampling is repeated for each node in the tree.

Therefore, sampling data to build each predictor of the ensemble results in different decision trees generated on slightly different data sets, while feature sampling at each node generates trees generated on different subsets of the feature vector. The combination of both sampling strategies produces a forest of different decision trees. Once training is complete, the random forest predicts targets using a soft voting strategy, as explained previously. An example of a random forest consisting of eight trees is depicted in Figure 2.20, where the classification boundaries of each predictor and the random forest are depicted in different colors. The decision boundaries of the seven trees are quite different and less intuitive compared to that of the random forest, which is smoother (more trees would create even smoother boundaries).

Gradient Boost *Boosting* refers to any ensemble method that combines several weak learners (simple models) to create a strong one. The basic idea behind boosting methods is to train predictors sequentially, so that each predictor corrects the previous predictor, i.e. a predictor becomes better by learning from the previous predictor's errors. This is achieved by fitting a new predictor to the residual errors made by the previous predictor (Figure 2.21). There are many boosting methods in the literature, one of which has recently dominated many machine learning competitions. The method is called gradient boosted regression trees (gradient boost), and despite the term regression in the name, it can be used for both regression and classification. Compared to random forest, gradient boosted regression trees most of the time outperforms it if its parameter are well tuned. This is actually the main disadvantage of gradient boosted decision trees - the sensitivity to parameter tuning, compared to other supervised techniques.

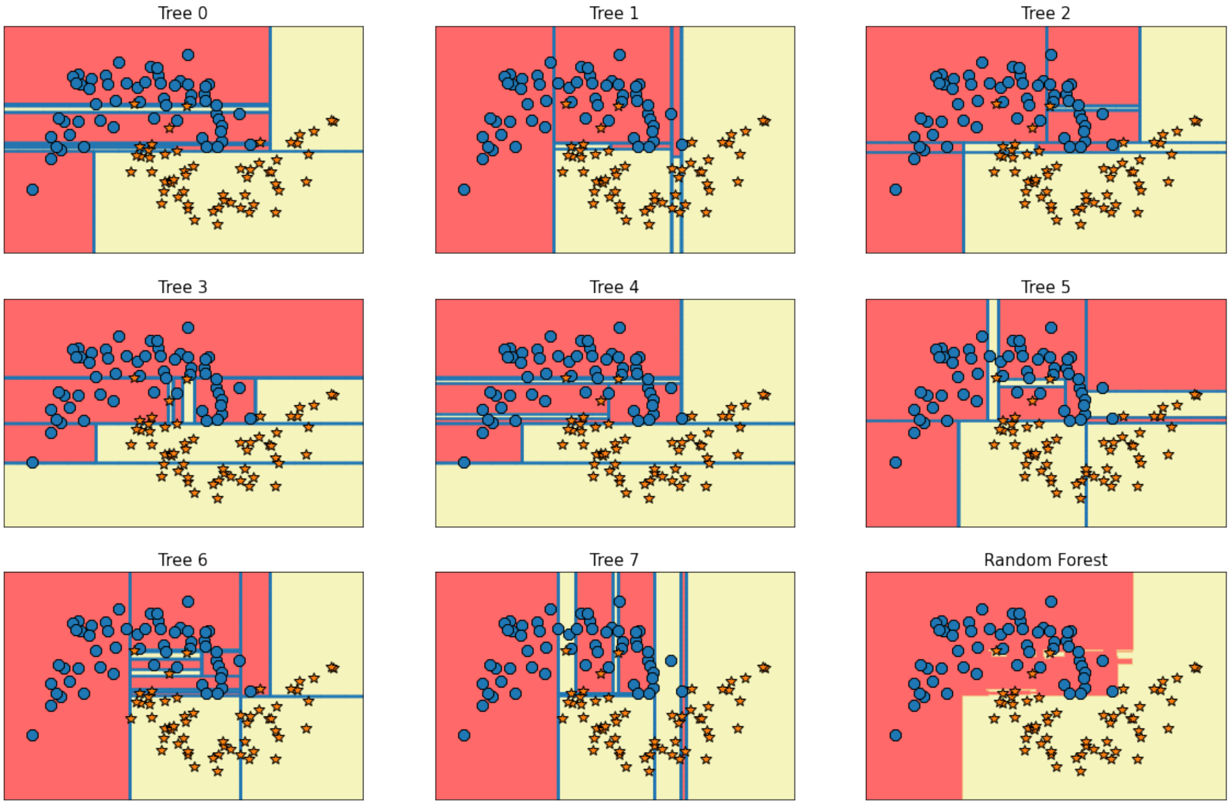


Figure 2.20.: A random forest (lower right figure) created from eight decision trees. Targets from different classes are indicated in different color and shape (balls and asterisks). Decision boundaries are illustrated in yellow and orange color.

The gradient boost method combines multiple low-depth decision trees, which are good predictors for a portion of the data, to create a more robust model. When constructing the trees, no randomness is introduced by default, but strong pre-pruning is applied to keep the trees shallow (usually the depth is limited between one and five), resulting in a fast predictor with low memory requirements. In addition to the number of trees (estimators) and the maximum depth that regulate the complexity of the model, another parameter of the gradient boost algorithm is the learning rate, which regulates the contribution of each tree to the residual error correction. A low learning rate requires more trees for the ensemble to fit the training data, but usually the generalization ability of an ensemble with more trees is better than when a higher learning rate and fewer trees are used.

A machine learning library that implements a scalable distributed gradient-boosted decision tree model is XGBoost, which stands for Extreme Gradient Boosting. For large-scale problems, it can be faster than other implementations (e.g., Scikit-Learn). Figure 2.22 illustrates the evolution of decision trees in random forest, gradient boosted trees and the latest implementation of the latter (XGBoost) that allows faster predictions.

2.3.2.3. Metrics for Classification

As explained earlier, a classifier is trained using a dataset called training dataset. After the model is trained, its performance must be evaluated in a rigorous manner. The evaluation of the model informs how well the model can predict on data that it has not seen before.

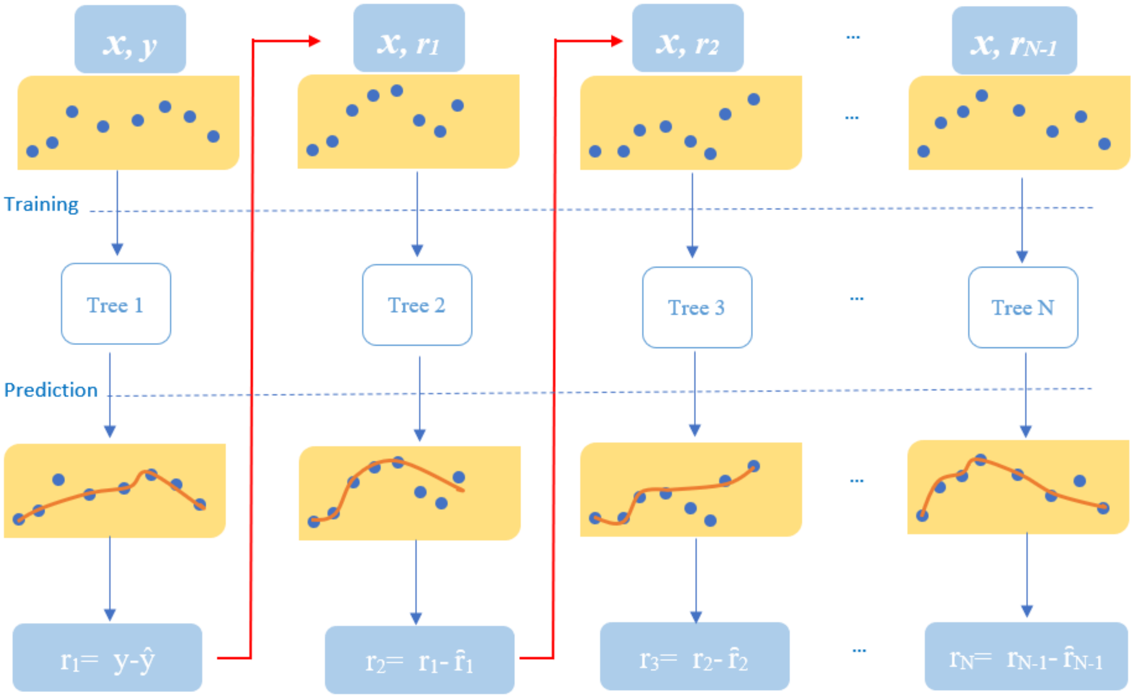
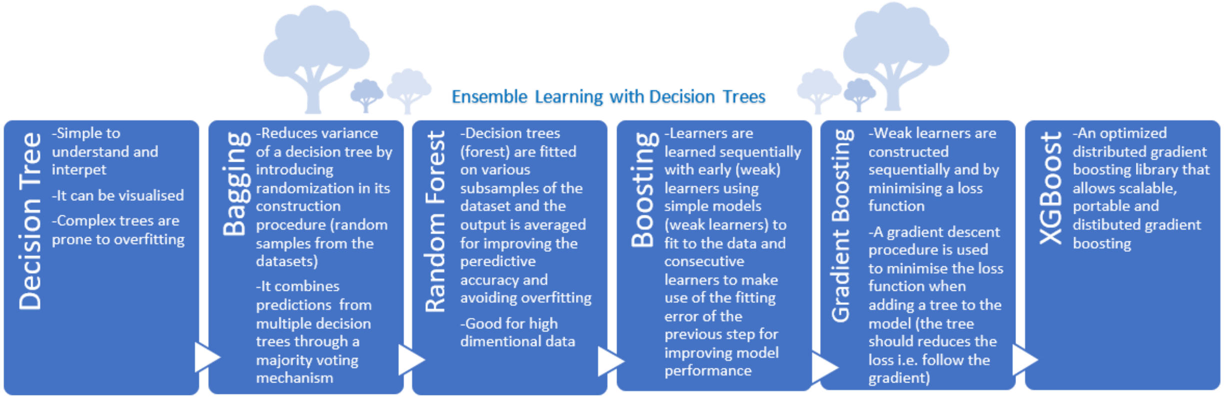


Figure 2.21.: The sequential process of building a gradient boosted decision tree model. New predictors are added iteratively, by training each new tree on the residual errors of the predecessor tree.


Ensemble Learning with Decision Trees

Decision Tree
-Simple to understand and interpet
-It can be visualised
-Complex trees are prone to overfitting

Bagging
-Reduces variance of a decision tree by introducing randomization in its construction procedure (random samples from the datasets)
-It combines predictions from multiple decision trees through a majority voting mechanism

Random Forest
-Decision trees (forest) are fitted on various subsamples of the dataset and the output is averaged for improving the peredictive accuracy and avoiding overfitting
-Good for high dimentional data

Boosting
-Learners are learned sequentially with early (weak) learners using simple models (weak learners) to fit to the data and consecutive learners to make use of the fitting error of the previous step for improving model performance

Gradient Boosting
-Weak learners are constructed sequentially and by minimising a loss function
-A gradient descent procedure is used to minimise the loss function when adding a tree to the model (the tree should reduces the loss i.e. follow the gradient)

XGBoost
-An optimized distributed gradient boosting library that allows scalable, portable and distibuted gradient boosting

Figure 2.22.: The evolution of a single decision tree to random forest, gradient boosted decision trees and the recent implementation of the latter for faster predictions (XGBoost library).

Such a dataset is called a test dataset. A good performance on the test dataset means that the model generalizes well, so learning through training has been successful. Some formal widely used performance metrics for multi-class classification are *confusion matrix, accuracy, precision/recall and f-measure.*

Confusion Matrix The confusion matrix is a table that summarizes the classification results of a classifier in the predicted classes. Each raw corresponds to an actual class and each column to a predicted class. The diagonal contains the examples that have been successfully predicted (the actual and predicted class are the same). For a binary classification problem, where the examples of the two classes are labelled as positive and negative, TP indicates the true positive classified examples, TN the true negative, FP the false positive, and FN the false negative examples. These values can then be compactly portrayed in the confusion matrix, as shown in Table 2.2.

Table 2.2.: A confusion matrix of a binary classification problem.

		predicted	
		class 1	class 2
actual	class 1	TP	FN
	class 2	FP	TN

The confusion table for multiclass classification is constructed in a similar way. It has as many rows and columns as there are different classes (e.g., N classes) and each cell c_{ij} with i,j $\in$ 1,2,...,N contains the number of examples from the actual class i, which are (mistakenly) assigned to class j. The examples that are correctly classified are found in the diagonal cells c_{ii}. TP, FP, TN and FN are defined with respect to each class. For a class i, all examples of class i are considered positive and all examples from other classes are considered negative. Often the rates of the aforementioned measures are used, such as the True Positive Rate (TPR) also called *recall* or *sensitivity*, True Negative Rate (TNR) also called *specificity*, False Positive Rate (FPR) and False Negative Rate (FNR), which are defined as:

$$\text{TPR} = \frac{TP}{TP + FN} = 1 - \text{FNR} \tag{2.3}$$

$$\text{TNR} = \frac{TN}{TN + FP} = 1 - \text{FPR} \tag{2.4}$$

$$\text{FPR} = \frac{FP}{FP + TP} = 1 - \text{TNR} \tag{2.5}$$

$$\text{FNR} = \frac{FN}{FN + TP} = 1 - \text{TPR} \tag{2.6}$$

Confusion matrices can be used to calculate other performance metrics, such as accuracy, precision, and recall.

Accuracy The accuracy of a predictor is defined by the number of correctly classified examples divided by the total number of examples (correctly and incorrectly classified).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.7}$$

Accuracy is a useful metric when prediction errors are equally significant across classes.

Precision/Recall The precision is the ratio of the correct classified positive samples (TP) to the overall positive classified samples (either correctly or incorrectly, i.e., TP and FP). It expresses therefore the proportion of positive predictions that was actually correct and reflects how reliable the model is in classifying samples as positive. It is defined as:

$$Precision = \frac{TP}{TP + FP} \tag{2.8}$$

Recall (also called sensitivity) is the ratio of correct classified positive samples to the overall number of actual positive examples in the dataset. It expresses what proportion of actual positive samples was predicted correctly and it actually measures the model's ability to detect positive samples. The higher the recall, the more positive samples detected. It is defined as:

$$Recall = \text{TPR} = Sensitivity = \frac{TP}{TP + FN} = 1 - \text{FNR} \tag{2.9}$$

F1 score F1 score is the harmonic average of recall and precision.

$$F1score = \frac{2}{\frac{1}{recall} + \frac{1}{precision}} = 2 * \frac{recall * precision}{recall + precision} \tag{2.10}$$

Macro average/weighted average scores In a classification report which is often exported after evaluating the performance of the test set, besides the aforementioned scores, are also printed the macro and weighted values of the these scores (macro average precision, weighted average precision, etc.). If P is the number of positive saples, N the number of negative in a binary classification problem, then these scores are defined as:

$$score_{macro\_avg} = \frac{1}{2} score_{P.} + \frac{1}{2} score_N \tag{2.11}$$

$$score_{weighted\_avg} = \frac{P}{P+N} * score_P + \frac{N}{P+N} * score_N \tag{2.12}$$

2.3.3. Unsupervised-Learning: Clustering

Unsupervised learning includes all kinds of machine learning where there is no known output to guide the learning algorithm. The goal of the learning algorithm is, given the (unlabeled) input data to analyze and discover knowledge from them. The knowledge can be in the form of hidden patterns or groupings of data (similarities). Unsupervised learning models are used for three main functions: clustering, association and dimensionality reduction. Here we focus only on clustering.

Cluster analysis partitions data into groups (clusters) of similar objects that are meaningful and therefore can be useful for certain applications. A set of clusters is usually referred to as a clustering. Clustering learning algorithms can be categorized into the following types: *hierarchical* (nested) versus *partitional* (unnested), *exclusive* versus *overlapping* versus *fuzzy*, and *complete* versus *partial* (Tan et al., 2018, p. 311). A partitional clustering splits the set of data instances into non-overlapping subsets (clusters) such that each data instance

Algorithm 2: The K-means algorithm.

```
Data:
D : a set of data
k : number of clusters
e : convergence criterion
Initialize cluster centroids $\mu_1, \mu_2, ..., \mu_k$ randomly
Initialize cluster sets to empty sets: $C_j \leftarrow \emptyset \; for \; all \; j = 1, ..., k$
repeat
    // Step 1: Cluster Assignment
    for $x_j \in D$ do
        $j \leftarrow arg\ min_i\{\|x_j - \mu_i\|\}^2 //assign\ x_j\ to\ closest\ centroid$
        $C_j \leftarrow C_j \cup \{x_j\} //add\ x_j\ to\ the\ relevant\ cluster\ set$
    end for
    // Step 2: Cluster Centroid Update
    for $i = 1\ to\ k$ do
        $\mu_i \leftarrow \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j$
    end for
until convergence criterion e is met;
```

belongs to exactly one cluster. If a cluster is allowed to have subclusters, then the clustering is called hierarchical, and since it consists of a set nested clusters, it is organized in a tree-like structure. An exclusive clustering assigns each data instance to a single cluster, while an overlapping or non-exclusive clustering allows data instances to be assigned to more than one cluster. Under fuzzy (or probabilistic) clustering, each data instance belongs to every cluster with a membership weight ranging between 0 and 1. Fuzzy clustering differs from overlapping clustering because the membership weights (probabilities) for each data instances sum up to 1 and therefore cannot deal with multiclass situations, where a data instance may “fully” belong to two classes. A complete clustering assigns each data instance to a cluster, while a partial clustering allows data instances to be assigned to no cluster. It is often the case that some data instances in the dataset are outliers or represent data that are not meaningful in the context of the similarities of the intended clustering of the data. These data instances under partial clustering are not assigned to any cluster.

2.3.3.1. The K-means Clustering Algorithm

K-means is a partitional clustering technique that tries to find a user-specified number of clusters (K), given as parameter to the learning algorithm. The clusters are represented by their centroids. The K-means algorithm, described formally in Algorithm 2, starts by initializing the K centroids. Then each data instance is assigned to the closest centroid. All data instances assigned to the same centroid belongs to the same cluster. The centroid of each cluster is then recomputed (update) based on the values of the data instances that are assigned to the cluster that the centroid represents. The data instance assignment and centroid recomputation steps are repeated until a convergence criterion is met. Usually such criterion is when the centroids remain the same or almost the same (no data instance changes cluster or a small percentage of them shifts to other clusters).

An example of the K-means algorithm is depicted in Figure 2.23. Here the predefined number of clusters that the clustering algorithm needs to group the data instances to, is three, and after initializing the centroids, three iterations of assigning all the points to clusters and updating the centroids based on the assignments to clusters are needed until the algorithm to converge.

K-means is simple, intuitive and can be used for a wide variety of data types. However, it is not suitable for clusters of varying size, shape form and density, although it can usually detect discrete small clusters if the user has specified a large number of clusters. Another weakness is the sensitivity to outliers, inherited due to the recalculation of centroids from the data instances and the subsequent reassignment of data instances into clusters based on proximity to centroids.

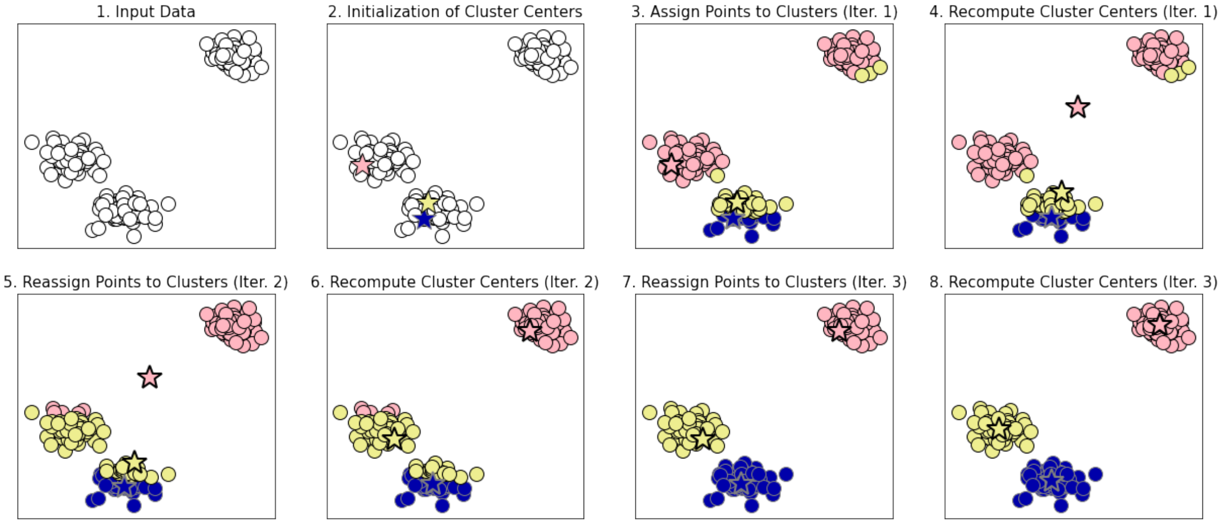


Figure 2.23.: An example of the K-means clustering algorithm, where data instances are assigned to three clusters (pink, yellow, blue), within three iterations of point assignment and centroid update. The centroids of clusters are indicated by asterisks.

2.3.3.2. DBSCAN

DBSCAN (Ester et al., 1996) is a partitional, partial, density-based clustering algorithm that does not require the user to specify the number of clusters to be identified. It is partitional because data instances are detected in non-overlapping regions and partial because low-density regions are classified as noise and assigned to no cluster. DBSCAN is defined formally in Algorithm 3.

DBSCAN is characterised (and named) as density-based clustering because it identifies high-density data regions separated by low-density regions. Here, the density is estimated around a particular data instance x and is defined by the number of data points ($min\_samples$), including x, that lie within a given radius eps of x. It is clear that the two parameters eps and $min\_samples$, which define the density, affect the cluster detection, as shown in Figure 2.24. For example, a large eps will consider distinct clusters as one (last column of figures), while a very small eps will split the data into very small clusters (first row, second figure from left).

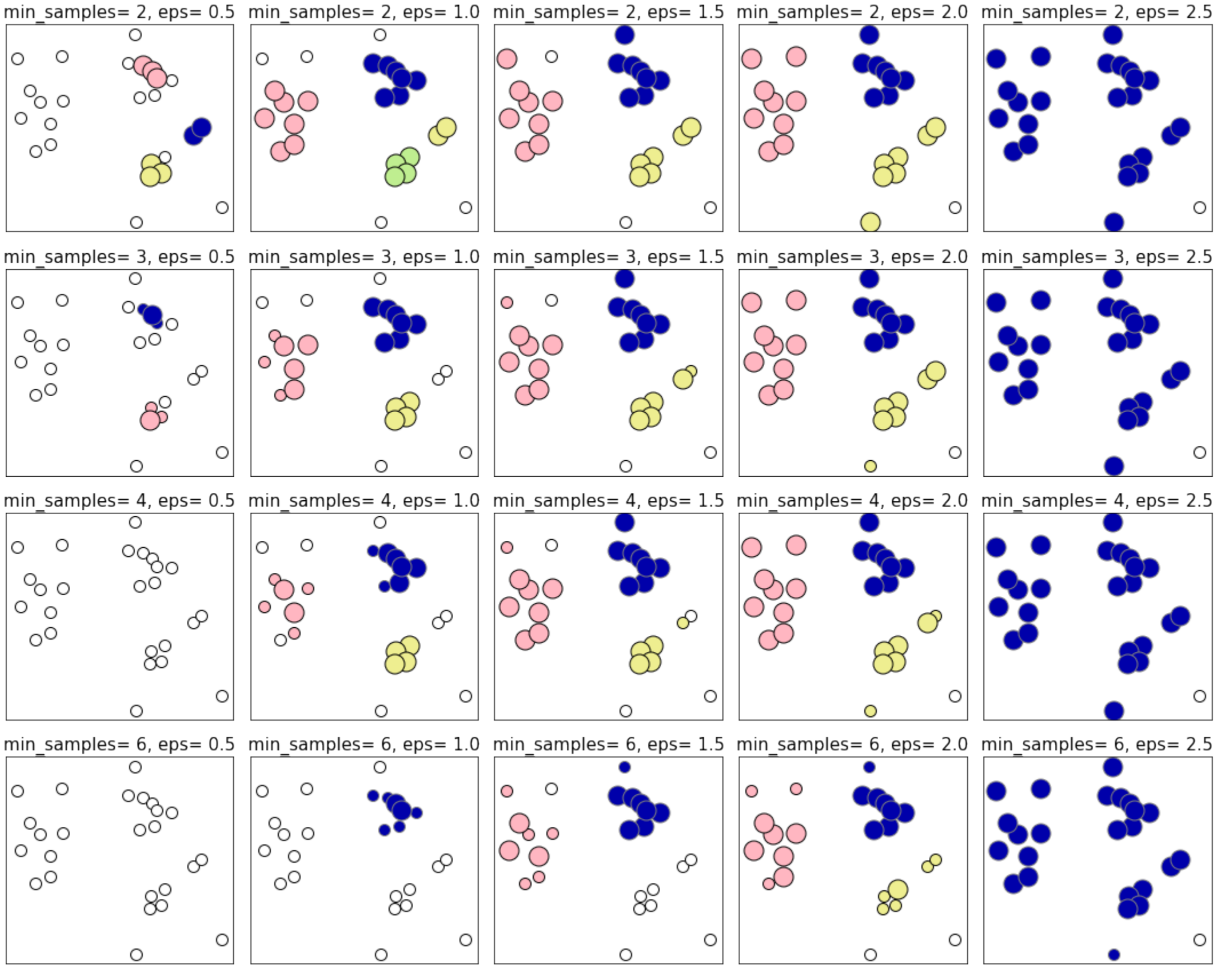


Figure 2.24.: The DBSCAN algorithm for different values of parameters.

To formalize whether a region in the data space is dense or non-dense, three concepts are defined: *core* point, *border* and *noise* point (Figure 2.25). Core points are located inside a dense region (cluster), and for a point to qualify as a core point, it must have at least *min_samples* in a radius of size *eps* (measured from the point). Border points are points that are not core points, but fall within the neighborhood of a core point (or multiple core points), as defined by *eps*. Noise points are any point that is neither a core point nor a border point. DBSCAN can handle clusters of arbitrary shape and size and is relatively noise efficient. However, when densities vary, DBSCAN fails to identify the correct clusters.

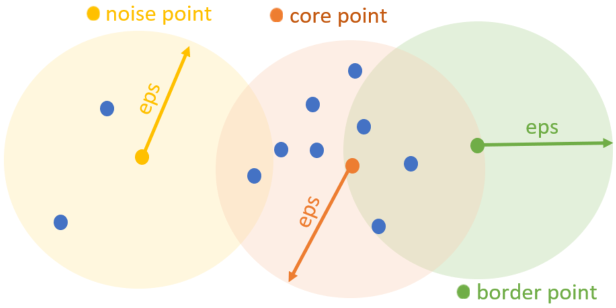


Figure 2.25.: Core, border and noise points (min_samples = 10).

Algorithm 3: The DBSCAN algorithm.

```
Data:
$D$ : a set of data
$eps$ : the radius defining the neighbourhood around a data point
$min\_samples$ : minimum number of points to define the region as dense
Initialize the set of clusters $C$ to an empty set: C $\leftarrow \emptyset$
Mark all data $x_i \in D$ as Unprocessed
foreach $x_j \in D$ do
    if $x_j$ is already Processed then
        continue to next data point
    end if
    else
        Mark $x_j$ as Processed
        neighbors $\leftarrow Eps\_neighborhood(x_j, eps)$
        if $|neighbors| < min\_samples$ then
            Mark $x_j$ as Noise
        end if
        else
            $C \leftarrow C_{new}$ //create a new cluster
            Call $Expand\_Cluster\_Function(x_j, neighbors, C_{new}, min\_samples)$
        end if
    end if
end foreach
```

Algorithm 4: Expand cluster function of the DBSCAN algorithm.

```
Data:
$x_j$ : a data point
$neighbors$ : neighbor points of $x_j$ with respect to $eps$
$C$ : a cluster
$eps$ : the radius defining the neighbourhood around a data point
$min\_samples$ : minimum number of points to define the region as dense
Add $x_j$ to cluster $C$
foreach $x_i \in neighbors$ do
    if $x_i$ is not already Processed then
        Mark $x_i$ as Processed
        $expand\_neighbors \leftarrow Eps\_neighborhood(x_i, eps)$
        if $|expand\_neighbors| < min\_samples$ then
            $neighbors \leftarrow neighbors \cup expand\_neighbors$
        end if
    end if
    if $x_i$ is not assigned to any cluster then
        Add $x_i$ to cluster $C$
    end if
end foreach
```

2.3.4. Semi-supervised Learning

Semi-Supervised Learning (SSL) addresses the situation where relatively few labeled training data instances are available, but a large number of unlabeled data is provided. In many practical problems, obtaining labeled data can be costly, for example for automatic web page classification and part-of-speech tagging (millions of labelled data are required), or difficult, such as in computer-assisted medical diagnosis applications. As a branch of machine learning, SSL uses a diverse set of tools developed in other branches of machine learning (unsupervised learning and supervised learning) and therefore lies midway between supervised and unsupervised learning (Chapelle et al., 2006, p.17). Formal SSL uses data $X = (x_i)_{i \in [n]}$ that can be divided into two parts. The data instances $X_l = (x_1,x_2,...x_l)$, for which the labels $Y_l = (y_1,y_2,...y_l)$ are given, and the data instances $X_u = (x_{l+1},x_{l+2},...x_{l+u})$ for which the labels are not known.

The SSL algorithms generally aim to improve performance on one of the two tasks related to supervised and unsupervised learning, using information generally related to the other. For example, semi-supervised classification methods attempt, by exploiting instances of unlabeled data, to create a classifier whose performance exceeds the performance of classifiers created using exclusively labeled data. This is motivated by an assumption called the *smoothness* assumption which states that, for two input data instances $x,x' \in X$ that are close to the input space, the corresponding labels y,y' should be identical. In practice, SSL methods have also been applied to problems where labeled data were not sparse. The assumption here is that if unlabeled data points provide additional information relevant to the prediction task, then they can potentially help improve classification performance. For clustering problems, the learning process may also benefit from the knowledge that some data instances belong to the same class (Van Engelen and Hoos, 2020). In this case the assumption is that data instances belonging to the same cluster belong to the same class too (assumption *cluster*).

The variety of SSL algorithms is quite large. The methods differ in the SSL assumptions on which they are based, in the way they incorporate unlabeled data into the learning process (how unlabeled data are selected), and in the way they relate to supervised learning algorithms. The most recent survey in SSL (Van Engelen and Hoos, 2020) proposes a classification of methods, which at the highest level are classified into two categories, *inductive* and *transductive* methods. Inductive methods try to find a classification model that can be used to predict the labels of previously unseen data instances, while transductive methods aim to obtain label predictions for unlabeled data instances without creating a learning model from the input space. Thus, given a dataset consisted of labeled and unlabeled data, $X_l,X_u \subseteq X$, with labels $y_l \in Y^l$, the inductive methods obtain a model from the l labeled data instances, $f : X \mapsto Y$, while the transductive methods obtain predicted labels $\hat{y}_u$ for the unlabeled data instances in X_u. At a second level, the taxonomy categorizes inductive methods into three classes depending on how they incorporate unlabeled data: either through a pseudo-labeling step (wrapper methods), a preprocessing step (unsupervised preprocessing), or directly within the objective function (intrinsically SSL methods). The transductive methods are in all cases based on graphs. In the following sections, we focus only on one simple wrapper method, *self-training*, and one preprocessing method, *cluster-then-label*, as these methods were used in the implementation of this thesis.

2.3.4.1. Self-Training (Self-Learning)

Self-training methods (also called *self-learning* methods) are the most basic of pseudo-labeling approaches. They consist of a single supervised classifier that is iteratively trained on both labeled data and pseudo-labeled data obtained in previous iterations of the algorithm (Van Engelen and Hoos, 2020, p. 385). Algorithm 5 lists the sequence of steps of the self-training process. Firstly, a classifier is trained with the available labeled data and then it makes predictions on the unlabeled data instances. From the predicted labels, the most confident ones (those predicted with high probability, greater than the confidence threshold defined as parameter in Algorithm 5) are called pseudo-labels and are added to the set of the labeled data. The training data after this addition consists of labeled and pseudo-labeled data. The classifier is then re-trained on the enlarged training dataset and the process is iterated until a stop-criterion is met. Usually, such a criterion is a maximum number of iteration or when all all unlabeled data has been labeled.

By using different conditions for the selection of pseudo-labels, several variants of self-training algorithms can be derived. Similarly, modifying the process of incorporating the selected pseudo-labeled data into the classification (how to reuse them, e.g., by giving them weights) or choosing different stopping criteria can lead to variations of self-training algorithms.

Algorithm 5: Self-training.

```
Data:
$L$ : the labeled data
$U$ : the unlabeled data
$stop–criterion$ : $k$ iterations or all unlabeled data $U$ has been labeled or the predictive
accuracy does not improve significantly anymore
$h$ : a classifier
$conf_{threl}$ : confidence threshold
repeat
    Train the classifier $h$ with the $L$ labeled data
    Predict the $U$ data with $h$ classifier
    Select the confident predicted data instances from the previous step, w.r.t $conf_{thres}$
      (pseudo-labels)
    Add the pseudo-labeled data instance to $L$
until stop–criterion is met;
```

2.3.4.2. Cluster-then-label

Both SSL methods considered so far exclusively use supervised learning to train classifiers. However, many SSL algorithms combine clustering with supervised learning, such that the former complements or guides the classification task. Such methods are called *cluster-then-label* approaches and in principle first apply an unsupervised or semi-supervised clustering algorithm on all available data, and then use the resulting clusters to “inform” the classification process.

One such example can be clustering all available data and then propagating the majority label of each cluster to all unlabeled data instances of the same cluster or to a percentage of data instances that are closest to the centroid of the cluster. Then, using the augmented labeled data (the original labeled data augmented by the clustering-assisted propagated labels), a classifier can be trained to predict the target labels.

Another variant of the cluster-then-label method first clusters the labeled data and a subset of the unlabeled data and then a different classifier for each cluster is trained using the labeled data contained in the cluster. The trained classifiers then predict the unlabeled data instances of the clusters whose labeled data were used to train them (Goldberg et al., 2009). Algorithm 6 describes the steps of an a cluster-then-label approach, where instances that are used for training by a classifier are selected in a earlier step with the help of clustering (Géron, 2019, p. 253).

Algorithm 6: Cluster-then-label.

```
Data:
$U_{train}$ : unlabeled data to be used for training
$U_{test}$ : unlabeled data to be used for testing (predicting)
$k$ : number of clusters
$h_1$ : a clustering algorithm
$h_2$ : a supervised classifier
$p$ : percentage of data instances (0-100)
Cluster the unlabeled data $U_{train}$ into $k$ clusters with $h_1$
for each cluster do
    Find the data instance $x_{closest}$ from each cluster that is closest to the centroid of the cluster
    Manual label the $x_{closest}$ data instance
    Propagate the label of $x_{closest}$ to $p$% of data instances that belong to same cluster as $x_{closest}$ and are closest to it
end for
Train $h_2$ with the data instances labeled in the previous step
Predict the test data $U_{test}$ using the trained $h_2$ classifier
```

2.3.5. Active-Learning

Active learning (sometimes called “query learning” or “optimal experimental design”) is a subfield of machine learning that focuses on how to select training data so that better learning, in terms of performance, is achieved with less training data. The basic assumption is that if the learning algorithm is allowed to choose the data from which it learns, it will perform better with less training (Settles, 2009). Unlike standard SSL methods that attempt to address the problem of sparsely labeled data by using supervised and unsupervised methods, active learning systems attempt to overcome the problem of label sparsity by asking queries in the form of unlabeled instances to be labeled by an *oracle* (e.g., a human annotator). With such a selection procedure, the *active* learning process aims to achieve better generalization performance with less training data, and thus at a lower cost.

There are several scenarios (Settles, 2009, p.8) in which data-driven active learners can pose questions in the form of unlabeled instances. One of these is called *pool-based sampling*, which assumes that there is a small set of labeled data L and a large pool of unlabeled data U. Then, queries to label unlabeled data are selectively drawn from the pool U, which is usually assumed to be static (unchanging). There are different query strategy frameworks in terms of the criteria for selecting the data instances to be labeled. One

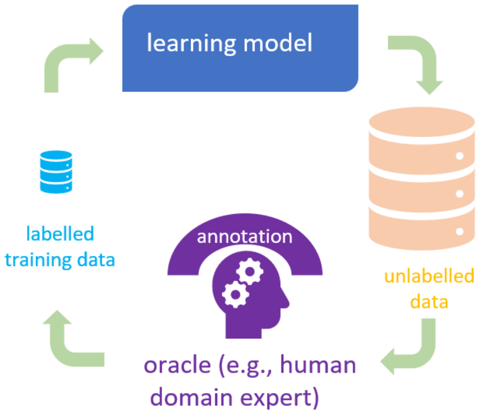


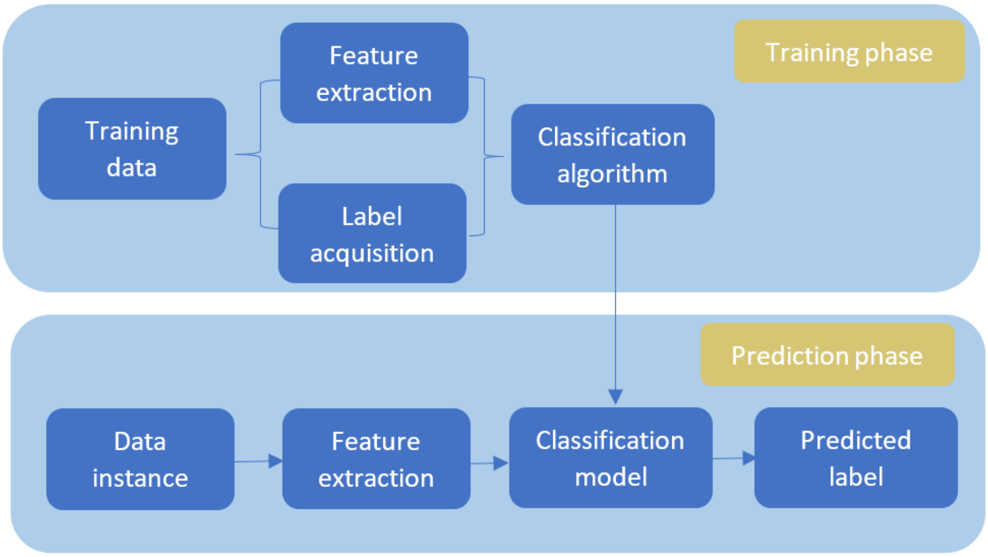
Figure 2.26.: The pool-based active learning process.

of them selects instances for which the classifier is less confident about their target label (uncertainty sampling). The learning process in a pool-based scenario is shown in Figure 2.26. Combined with an uncertainty sampling the learning steps are given in Algorithm 7. The learning model is trained on the labeled instances L and then used to make predictions on all unlabeled data instances U. The data instances for which the model is most uncertain (i.e., the estimated probability is the lowest) are labeled by a domain expert. This process is repeated until a stopping criterion is met (e.g., all data in the U pool are labelled). Other sampling strategies include labeling the data instances that would lead to the largest model change or the instances for which different classifiers disagree most. Algorithm 6 can be seen as an example of *hybrid* active learning (Lughofer, 2012), which is based on an unsupervised criterion followed by a supervised criterion based on uncertainty. Lughofer (2012) implemented such an active learning strategy in an on-line classification scenario, where in a first step training is conducted from scratch (i.e., no initial labels/learners are required) based purely on unsupervised criteria obtained from clusters. Samples located near cluster centres and near cluster boundaries are considered to be the most informative in terms of the characteristics of the class distribution and are therefore selected to be labeled. In the second step, the classifier is trained incrementally (on-line mode) using the most important data examples for training, selected in the previous (off-line) step.

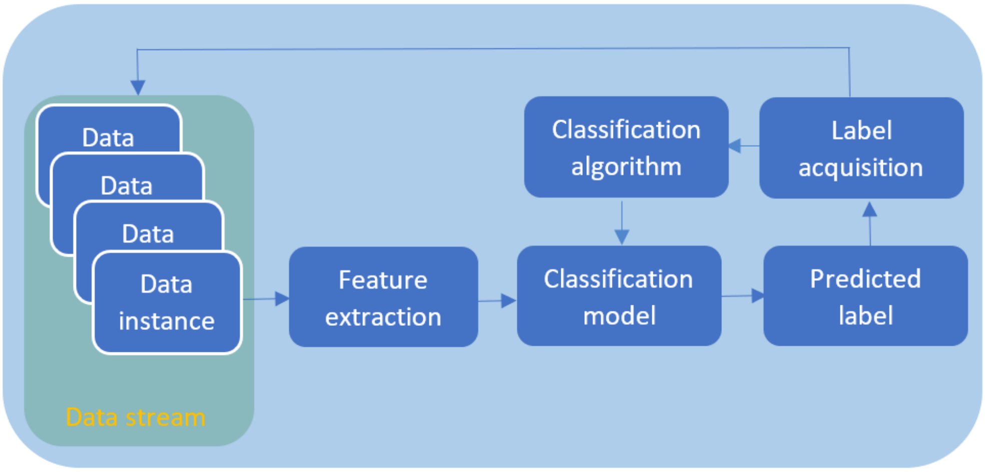
```
Algorithm 7: Active-Learning.
    Data:
    $U$ : unlabeled data
    $L$ : labeled data to be used for training
    $h$ : a supervised classifier
repeat
    Train $h$ with the labeled data instances $L$
    Apply $h$ on the unlabeled data $U$
    Find the data instance $x_j \in U$ for which $h$ is less certain for its label
    Label $x_j$
    Add the labeled data instance $x_j$ in $L$ : $L \leftarrow \mathrm{L} \cup \mathrm{x}_j$
    Remove $x_j$ from the pool $U$: $U \leftarrow \mathrm{U} \setminus \mathrm{x}_j$
until a stop criterion is met;
```

2.3.6. Incremental Learning

All the learning processes discussed so far have two distinct stages, training and predicting. In the training phase, the stored data is processed by a learning algorithm that results in a model, which can then be used to make predictions on unseen data. This learning process is called batch or offline learning and is illustrated in Figure 2.27a. However, nowadays there are real-world applications where the data "ages" very quickly (old data may not represent the current state of the object or phenomenon under observation) or the amount of data to be processed is huge, and aged data cannot be stored (data may exceed the available memory). For example, forecasting applications, such as stock market forecasts, require real-time or near real-time processing, as its data source is non-stationary. Data that has the characteristic of being continuously generated from sources is called *stream data*, and in the context of machine learning it must be processed one observation at a time (or mini-batch), as opposed to batch learning, where learning is done in a batch manner. This learning process is called *incremental* or *online* and is illustrated in Figure 2.27b.



(a)



(b)

Figure 2.27.: (a) Batch learning, (b) Incremental (online) learning.

In the context of data stream classification, data instances are not available in the form of a set of numerous instances that can be used for training, but they become readily available in a continuous and fast way. Prediction requests are expected any time and the classifier must use its current model to serve them. Another characteristic of the data streams is that the statistical properties of the target variable, which the learning model predicts, or the input itself (more rare) can change over time. This characteristic is called *concept drift*, and it is clear that if the current model does not evolve or is not adjusted to capture such changes, predictions will gradually stop being accurate. Therefore evolving data streams dictate different learning settings than the typical batch learning, which have motivated modifications to current learning algorithms or new ones that meet those requirements. One classification algorithm that has been modified to meet the requirements of data streams is the *adaptive random forest* (Gomes et al., 2017). Algorithm 9 shows in pseudocode the random forest for stream data, and Algorithm 8 the training process of a random tree.

The adaptive random forest differs from the random forest in two ways: (1) in the way the bootstraps are created (online bootstrap procedure, line 2 of the Algorithm 8, (Bifet et al., 2010)) and (2) in the way the decision to split leaves is made (features are eliminated up to $m < M$, with M representing all classification features. See the arguments with which the RFTreeTrain function is called in Algorithm 9, line 9). In addition, splitting attempts (line 7, Algorithm 8) are restricted to these m features for that node. Smaller values of *Grace Period* parameter GP (line 6, Algorithm 8) motivate node splitting and lead to deeper trees. Up to line 11 (Algorithm 9), the algorithm addresses stationary data streams. The conditions tested in lines 11 and 16 attempt to address the problem of concept drift, where a drift warning (line 11) triggers the initialization of a background tree to capture the new "concept", which will only replace an existing tree if the drift is eventually confirmed (line 16 Algorithm 9).

Algorithm 8: RFTree Train of Adaptive Random Forest algorithm (Gomes et al., 2017).

```
   Data:
   $\lambda$ : fixed parameter of Poisson distribution
   $GP$: Grace period before recalculating heuristics for split test
   $m$: maximum features evaluated per split
   $t$: tree
   $x$,$y$ : data instance
1  RFTreeTrain(m,t,x,y)
2      $k \leftarrow Poisson(\lambda = 6)$
3      if $k > 0$ then
4          l$\leftarrow FindLeaf(t,x)$
5          UpdateLeafCounts(l,x,k)
6          if InstanceSeen(l) $\geq GP$ then
7              AttempSplit(l)
8              if DidSplit(l) then
9                  CreateChildren(l,m)
10             end if
11         end if
12     end if
```

Algorithm 9: Adaptive Random Forest (Gomes et al., 2017).

```
Data:
  $m$ : maximum features evaluated per split
  $n$ : total number trees ($n = |T|$)
  $\delta_w$ : warning threshold
  $\delta_d$ : drift threshold
  $c(\cdot)$: change detection method
  $S$ : data stream
  $B$ : set of background trees
  $W(t)$ : tree $t$ weight
  $P(\cdot)$ : learning performance estimation function
$T \leftarrow$ CreateTrees(n)
$W \leftarrow$ InitWeights(n)
$B \leftarrow \emptyset$
repeat
    (x,y)$\leftarrow next(S)$
    for all $t \in T$ do
        $\hat{y} \leftarrow predict(t,x)$
        $W(t) \leftarrow P(W(t),\hat{y},y)$
        $RFTreeTrain(m,t,x,y)$ //Train t on the current instance (x, y)
        //Check if Warning has been detected
        if $C(\delta_w,t,x,y)$ then
            $b \leftarrow CreateTree()$// Init background tree
            B(t) $\leftarrow$ b
        end if
        //Check if Drift has been detected
        if $C(\delta_d,t,x,y)$ then
            $b \leftarrow CreateTree()$ //Replace t by its background tree
            $t \leftarrow B(t)$
        end if
    end for
    //Train each background tree
    foreach $b \in B$ do
        $RFTreeTrain(m,b,x,y)$
    end foreach
until $HasNext(S)$;
```

2.4. Acknowledgements

Figure 2.15 has been published in the 22nd AGILE Conference on Geo-information Science (Zourlidou et al., 2019). Figure 2.22 has been published in the 25th AGILE Conference on Geographical Information Science (Zourlidou et al., 2022d). The thesis author is the principal investigator and author of these articles, as well as the creator of all figures.

3. Related Work

3.1. Existing Traffic Regulation Recognition Approaches

This thesis proposes a new classification of traffic regulation recognition studies according to the features used to classify traffic regulations. This taxonomy at the first level distinguishes three categories, namely *static*-, *dynamic*- and *hybrid*-approaches and at the second level five categories: *map-based*, *image-based*, *episode-based*, *speed-profile*, and *movement summarization*. The taxonomy is depicted graphically in Figure 3.1. All the studies reviewed in this section are listed in Table 3.1. A detailed description of the methods of the reviewed articles, their limitations and a comprehensive critical overview of the research field can be found in a related systematic literature review (Zourlidou and Sester, 2019), the first and so far only attempt to illustrate the progress and challenges of the research field. The review provided in this chapter, is extended to include recent works published after 2019.

This chapter provides a brief updated review of existing methods, focusing mainly on the following aspects and experimental settings in which they have been tested: the classes of regulations for which classifiers are trained to recognise, the characteristics of the datasets on which the methods are tested (are they openly available for download for reproduction of results or for use in testing similar methodologies? how was the groundtruth map conducted?), the testing settings (does the article consider the transferability of classifier learning to different cities?[1], does it examine classification performance using different numbers of GPS trajectories?), the classification features they use and their classification performance. This extracted information from the thirteen reviewed articles is organized in Table 3.2. By reviewing the existing methods, the knowledge gap that this thesis attempts to fill is highlighted.

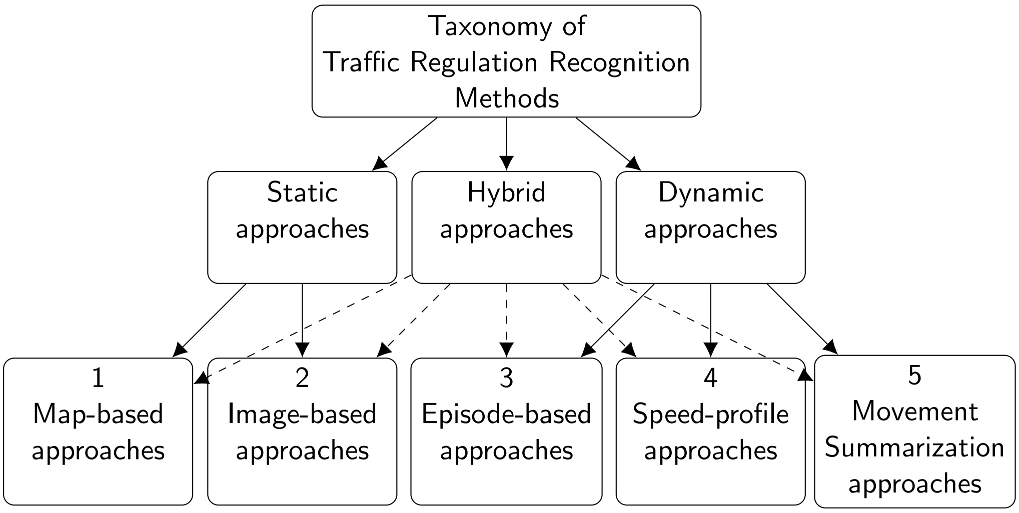


Figure 3.1.: Taxonomy of methods for traffic regulation recognition from GPS data.

---

[1] Cross-city learning transferabilty: train a classifier on the data of a city A and then predict target labels from data of a city B.

Table 3.1.: Reviewed articles in chronological order.

	Author(s)	**Title of the Article**	**Year**	Method(s)
1	Pribe and Rogers	Learning to associate observed driver behavior with traffic controls	1999	Episode/Mov.sum
2	Carisi et al.	Enhancing in vehicle digital maps via GPS crowdsourcing	2011	Episode
3	Saremi and Abdelzaher	Combining map-based inference and crowd-sensing for detecting traffic regulators	2015	Map, Epis., Hybrid
4	Hu et al.	SmartRoad: Smartphone-based crowd sensing for traffic regulator detection and identification	2015	Episode
5	Aly et al.	Automatic rich map semantics identification through smartphone-based crowd-sensing	2017	Episode
6	Wang et al.	Automatic intersection and traffic rule detection by mining motor vehicle GPS trajectories	2017	Mov.sum.
7	Efentakis et al.	Crowdsourcing turning restrictions from map-matched trajectories	2017	Mov.sum.
8	Munoz-Organero et al.	Automatic detection of traffic lights, street crossings and urban roundabouts combining outlier detection and deep learning classification techniques based on GPS traces while driving	2018	Speed
9	Zourlidou et al.	Classification of street junctions according to traffic regulators	2019	Speed
10	Méneroux et al.	Traffic signal detection from in-vehicle GPS speed profiles using functional data analysis and machine learning	2020	Speed
11	Golze et al.	Traffic regulator detection using GPS trajectories	2020	Episode/Mov.sum
12	Liao et al.	Impact assessing of traffic lights via GPS vehicle trajectories	2021	Speed/Map
13	Cheng et al.	Traffic control recognition with an attention mechanism using speed-profiles and satellite imagery data	2022	Speed/Image

Table 3.2.: Extracted information from the reviewed articles.

	Article	**Regulator***	**Open Data**	**Dataset Size**	**Ground-truth**	**Cross-city**	**Min. Samp.**	**Class. Method**	**Class.Perform.◇**
1	Pribe and Rogers (1999)	TS, SS, UN	No	50 Inter.	Various	No	No	Neural Nets	Ac:100%
2	Carisi et al. (2011)	TS, SS	No	89 Inter.	On site	No	Yes	Heuristics	Ac:>90%
3	Saremi and Abdelzaher (2015)	TS, SS, UN	No	>1K Inter.	Google Str.	Yes	No	Random Forest	Ac:91%(dyn), Ac:95%(map), Ac:97%(hyb)
4	Hu et al. (2015)	TS, SS, UN	No	463 Arms	On site	No	No	Random Forest, Spectral Clust.	Ac:>90%(RF), Ac:80%(Clust), Ac:95%(active learning)
5	Aly et al. (2017)	TS, SS	No	24km Foot Traj.	On site	No	No	Heuristics	R.:0.8, Pr:0.8
6	Wang et al. (2017)	TR	No	321 Inter.	On site	No	No	Clustering	-
7	Efentakis et al. (2017)	TR	No	>100K Inter.	Various	No	No	Heuristics	Ac:57–70%
8	Munoz-Organero et al. (2018)	TS**	Partly	55 Traj.,10 Traj	On site	No	No	Deep Belief Net.	R:0.89, Pr:0.88
9	Zourlidou et al. (2019)	TS, PS-YS	No	31 Inter.	On site	No	No	C4.5	R:0.83, Pr:0.31
10	Méneroux et al. (2020)	TS	No	44 TS, 5 SS	On site	No	No	Random Forest	Ac:95%
11	Golze et al. (2020)	TS, PS, UN	No	1064 Arms	On site	No	No	Random Forest	Ac:90.4%
12	Liao et al. (2021)	TS	No	-	On site	No	No	DLSTM	0.95 AUC
13	Cheng et al. (2022)	TS, PS, UN	No	3538 Arms, 1204 Traj.	On site	No	No	CVAE	F:0.90

Article: Authors of the article. Regulator: Regulator classes that are the targets of the learning/predicting process.

Open Data: indicates whether the dataset used in the study is openly available for download. Dataset Size: contains information on the size of the data if it is given by article. Groundtruth: indicates how the groundtruth map was conducted.

Cross-city: indicates whether the transferability of learning between cities is tested.

Min. Samp.: indicates whether the article conducts experiments on the effect of the number of trajectories on classification performance.

Class. Method: classification method. Class.perform.: classification performance.

* UN: Uncontrolled, TS: Traffic Signal, SS: Stop Sign, TR: Turning Restriction, PS: Priority Sign, YS: Yield Sign.

** Additionally to TS, street-crossings and roundabouts are detected.

◇ Ac: Accuracy, Pr: Precision, R: Recall, F: F-score, AUC: Area under the ROC Curve.

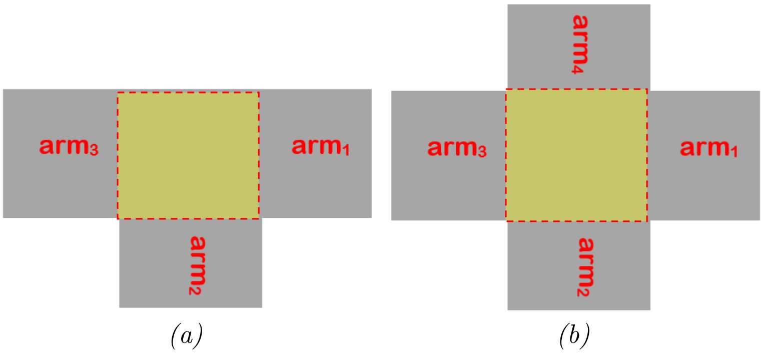


Figure 3.2.: Intersection arms of (a) a three-way and (b) a four-way intersection.

Here we should note that 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* (or *approaches*). The term intersection arm refers to the road that connects one intersection with another one, as explained on page 19 in Section 2.1.1. A three-arm (three-way) intersection has three arms and a four-way intersection has four arms, as shown in Figure 3.2a and 3.2b respectively.

The first level of taxonomy classifies methods into *static, dynamic* and *hybrid* categories, depending on whether the classification features they use are static, dynamic or a mixture from static and dynamic features. Static features are those that do not change over time, or if they do, they do so not very often. Such information can be extracted from maps (*map-based approaches*, Section 3.2.1) or satellite imagery (*image-based* approaches, Section 3.2.2). In contrast, dynamic features 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. In the dynamic category, we distinguish three subcategories: *episode-based* approaches (Section 3.3.1), where movement episodes such as stopping or deceleration events are used as classification features, *speed-profile* approaches (Section 3.3.2), and *movement summarization approaches* (Section 3.3.3).

3.2. Static Categorization

This section reviews methods from the two subcategories of static category, *map-based* (Section 3.2.1) and *image-based* approaches (Section 3.2.2).

3.2.1. Map-based Category

Map-based methods use static classification features extracted from maps, such as OSM. The intuition behind this idea is that features describing for example the connectivity of an intersection with nearby intersections, such as the distances of an intersection from neighboring intersections or the road category to which an intersection belongs (primary, secondary road, etc.) or the speed limit can be indicative of the regulation by which the intersection is controlled (see the considerations of intersection design explained on page 20).

Here we find only one study that uses such features (Saremi and Abdelzaher, 2015). They extract features related to speed as well as to inter-junction distances. 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.). After the classification, the methodology includes a step where a consistency check is done among the predicted labels on intersection level. The following domain knowledge rule is employed:

> *Either all or none of the approaches contributed 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, with 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 probabilities computed based on the fraction of decision trees voting for the approaches' alternative labels.*

The experiments were applied to datasets collected from four cities: 3,691 intersection approaches from the city of Urbana, 2,803 approaches from the city of Champaign, 7,561 approaches from Los Angeles, and 1,032 approaches from Pittsburgh. The groundtruth maps of the traffic regulations were created from street level images acquired from Google Street View. No dataset is available as an open dataset for reproducing the results or use for testing similar methods. 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). The cross-city learning transferability tests achieved an accuracy of 92%. No tests were conducted on the effect that the number of trajectories from which classification features are extracted may have on classification performance.

3.2.2. Image-based Category

Methods that use features extracted from satellite images are classified into the *image-based* category. Since this thesis focuses exclusively on methods that use GPS tracks, this category is only considered in the context of methodologies that, in addition to features extracted from tracks, also use features extracted from satellite images. Such a methodology is that of Cheng et al. (2022), which is discussed in the section of *hybrid* methodologies (3.4).

3.3. Dynamic Categorization

3.3.1. Episode-based Category

The third category, *episode-based*, includes approaches that use various features mainly related to stopping and/or deceleration episodes, usually extracted from a large set of trajectories (dynamic entities). 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 the same intersection arm, such as the average number of stopping episodes, the minimum number of deceleration episodes observed on tracks crossing an intersection, etc.

Carisi et al. (2011) proposed a heuristic method for solving a binary classification problem (traffic signals and stop signs), explaining how to enrich digital maps with the location and time of traffic regulator types. First, slow down and standstill episodes occurring near intersections are identified. Each intersection is then examined in terms of its traffic control.

An intersection arm (approach) is considered to be controlled by a potential stop if at least SST of the traces slows down. The SST is defined as the Stop Sign Threshold and its optimal value is defined to 80%. The information per intersection arm shall be aggregated to include all approaches of the same intersection. At an intersection regulated by stop signs, at most two ways could have the right of way, while all others must yield. Therefore, if all or all but one ways belonging to a given intersection, are marked as potential stops, the intersection is identified as stop-sign regulated, and all the ways marked with a potential stops become actual stop signs. If all but two ways are marked as potential stops, the algorithm is not able to make an immediate decision and performs an additional control loop to consider the case where the intersection is regulated with traffic lights. An intersection approach is considered as a potential traffic light if at least TLT of the vehicles approaching the intersection comes to a stop. The TLT is defined as the Traffic Light Threshold and its optimal value is set at 15%. Based on the observation that when an intersection is traffic light regulated, all its incoming ways comply with this requirement, the authors consider an intersection to be traffic light regulated when at least half plus one of the incoming ways are marked as potentially traffic light regulated. In addition, an intersection is marked as traffic light regulated, only if all the ways not classified by the stop-sign recognition algorithm meet the requirements to be potentially traffic signals. In all other cases, the intersection shall be classified as a two-way stop. After the intersections are classified as stop/traffic light controlled, the red signal duration is estimated using the 95th percentile of the standstill times. This methodology was tested on two sets of GPS traces from two sections of the city of Los Angeles (the 1st dataset consists of 28 intersections, 25 of which are stop signs and 3 traffic lights, and the 2nd dataset consists of 61 intersections, 25 regulated with stop signs and 36 traffic lights) and was found to be able to recognise stop signs with only 5 trajectories per intersection approach and traffic lights with only 7 trajectories, with an accuracy greater than 90%.

Saremi and Abdelzaher (2015) 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. As 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 stops 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. The dataset used to test this model consists of 6,700 miles of vehicular GPS traces collected from a total of 46 individuals over the course of several months. The minimum, first, second and third quartiles, and maximum number of trajectories per intersection approach covered by the GPS traces in the cities of Urbana, Champaign and Pittsburgh are 1, 2, 5, 24, 189, 1, 2, 4, 15, 223 and 1, 3, 5, 6, 13, respectively. It is not clear whether the number of intersections used to test the map-based model (Section 3.2.1) is the same for this model, as no detailed classification report is provided in the article. Also, 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% and interestingly, the accuracy under cross-city learning settings is found 93%.

Hu et al. (2015) use two types of classification features, *physical* and *statistical.* The physical features include the duration of the last stopping episode before a vehicle passes the intersection, the minimum passing speed, the number of vehicle deceleration episodes, the number of stopping episodes, and the distance of the last stopping episode from the intersection. The statistical features include the minimum, maximum, mean and variance of the physical features, aggregated for an intersection from all trajectories crossing it. The Random Forest classifier as well as Spectral Clustering were tested for a 3-category classification problem (traffic lights, stop signs, and uncontrolled intersections) using a dataset consisting of 463 intersection approaches (77 stop signs, 228 uncontrolled arms, and 158 traffic lights) crossed by 4,000 miles trajectories recorded by 35 volunteer participants. 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 junctions where trajectories always have to turn, will be always excluded entirely from classification. For this reason the following domain knowledge rule is applied, for labeling those arms too:

> *Let us denote the left, right, and bottom ends of a T-shape intersection A, B, and C, respectively. The traffic coming from C into the intersection must always turn, and thus will always be discarded. Then, if A/B-traffic is uncontrolled, C-traffic is stop sign controlled; otherwise, C-traffic has the same control type as A/B-traffic. In other words, if A/B has traffic lights, so does C; otherwise C has stop sign.*

The groundtruth map was conducted by on site observation. The accuracy is reported at 80% when unsupervised classification (clustering) is used. Under various feature settings and information aggregation schemes (data, feature, label aggregation) and using a Random Forest classifier, an accuracy of about 90% is reported. In addition, the paper considers an active learning framework, combined with self-training, where manual labeling requests are made to human annotators for those data instances predicted with low confidence. In this case, the accuracy reaches 95%, while with only 20% data for training, the classifier achieved 90% accuracy.

Aly et al. (2017) use pedestrian tracks to detect stop signs and traffic lights. They identify locations where pedestrians stay over a time interval (dwell time) and categorize the regulators accordingly. In particular, they measure the dwell time on the sides of crosswalks and if the median dwell time exceeds a threshold h_{dwell}, they identify the regulator as a traffic light. Otherwise, it is a stop signal. The method is applied to a dataset consisting of 24 km of pedestrian trajectories. The precision and recall of the classification is reported with respect to h_{dwell} values. For a relatively low value h_{dwell}, the precision and recall have optimal values, i.e., both 0.80.

3.3.2. Speed-profile Category

The *speed-profile* category includes methods that use vehicle speed profiles (time-ordered speed measurements) as classification features.

Munoz-Organero et al. (2018) detect traffic lights, road crossings and roundabouts in real time by classifying speed and acceleration time series with a deep belief network. Although, the combined recall and precision are relatively high (0.89 and 0.88 respectively), the score of the traffic light category shows a clear limitation in all tested classification settings, compared to the other two categories. Two small datasets were used to evaluate the method. One consists of 55 trajectories repeating the same route in two urban areas and a connecting highway (8.1 km), crossing various road elements, such as traffic lights, road crossings and roundabouts. The number of these elements is not given. The second dataset is an open dataset, consisting of 10 trajectories repeating the same 23.6 km long route, crossing the interesting locations: 2 ramps, 2 motorway exits, 2 roundabouts, 20 traffic lights and 2 curved roads. Despite the classification limitation of the traffic light category, this study is the first to use deep learning by taking into account the sequential order of speed samples within speed profiles, unlike other speed-based methods where the order of speed measurements is lost when fed to non-sequential classifiers (e.g., that of Zourlidou et al. (2019)).

Similarly, Zourlidou et al. (2019) investigate the *predictive* ability of speed profiles, both in terms of spatial resolution (speed sequences sampled at one-meter intervals) and temporal resolution (speed samples taken at one-second intervals). Different sequence sizes were tested, for the two types of resolution (31 samples, 51 samples, etc.), revealing that the classification performance was better when using a temporal resolution of 8 s from the centre of the intersection (a sequence of 9 speed samples, i.e., a speed sample 8 s from the centre of the intersection, in 7 s, 6 s, ..., 0 s, with 0 s corresponding to the time the vehicle crosses the centre of the junction). The speed profiles were used as features for a C4.5 decision tree classifier trained to distinguish between traffic signal controlled and priority/yield controlled intersections. The method was applied to a small dataset consisting of 31 intersections, 25 priority/yield controlled and 6 traffic signal controlled intersections. The results show high recall (0.83) for traffic light category prediction, but low precision (0.31) and F-measure (0.45). The groundtruth map of the dataset was conducted by on site observation and the trajectory dataset is not openly available.

Méneroux et al. (2020) detect traffic signals (binary classification problem) using speed profiles. By testing three different ways of feature extraction - functional analysis of speed records, raw speed measurements and image recognition technique - they found that the functional description of speed profiles with wavelet transforms outperforms the other approaches. Random Forest classification achieved the best accuracy (95%) compared to other tested classification techniques. However, as the authors point out, the lack of data is a strong limitation of the experiments, as their dataset contained only 44 instances of traffic lights.

3.3.3. Movement-summarization Category

The fourth categorization of methods is called *movement-summarization*. It uses as features the percentage of samples (trajectories) that follow specific motion patterns. The studies of Pribe and Rogers (1999) and Golze et al. (2020) use such features, but in combination with other features related to stopping/deceleration episodes, while the studies of Efentakis et al. (2017) and Wang et al. (2017), which detect turn restrictions, are based solely on such movement-summarization features.

Pribe and Rogers (1999) trained a Neural Network (NN) to learn to associate driver behavior with three types of traffic rules, traffic lights, stop signs, and clears (uncontrolled intersection arms). The NN was fed with the mean and standard deviation of the features associated with the stopping episodes extracted from the GPS traces. Specifically, the number of times a vehicle stops before crossing an intersection, the duration of the last three stops closest to the intersection, and the total duration of all stops were used. In addition, they calculated the percentage of trajectories that include at least one stop along each intersection approach. The method achieved an accuracy of 100%, but was tested on a rather small dataset (50 intersections), also ignoring intersections that did not have at least two arms crossing the tracks.

Golze et al. (2020) proposed a Random Forest classifier with oversampling and Bagging Booster to predict intersection regulators (traffic signals, priority controlled and uncontrolled intersections) with 90.4% accuracy. Along with other physical characteristics such as the number of stopping episodes, the duration of stopping events, the average distance to the intersection center of all stopping events, the duration of the last stopping episode, the distance to the intersection center of the last stopping event, the average speed and the maximum speed on approaching the intersection, they also calculate the percentage of trajectories with at least one stopping episode. By also performing 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. The dataset consists of 700 trajectories from the city of Hanover, Germany, each of which has a length between 5 and 14 km, with a total length of 3,748 km and all trajectories crossing a total of 1,064 intersections. The dataset is not openly available for download, and ground truth map has been carried out by on site observation.

A different category of rules concerns turn restrictions. Wang et al. (2017) detect turn-related rules, such as U-turns and left-turn prohibitions within a certain time range (time range that turns are allowed in specific intersection ways) by analyzing the trajectories and identifying turn patterns (if any and within what time range) using clustering. This trajectory analysis results in a table where rows represent different time intervals and columns correspond to intersection turn paths. Each cell then contains the number of trajectories that cross that turn path in a given time. Cells with zero trajectories indicate turn prohibitions.

Similarly, Efentakis et al. (2017) identify turning restrictions from a set of map-matched trajectories by detecting the collective turning behavior of vehicles at intersections, i.e., summarizing the movement behavior (turning or not) of vehicles at intersection locations. The study uses three OSM datasets collected from Athens, Berlin and Vienna, each containing thousands of turns on intersection approaches (75,552 in the Athens dataset, 44,636 in Berlin and 36,484 in Vienna). Using two different thresholds (2.5% and 5%), turning restrictions are identified by examining the percentage of trajectories turning on a given intersection approach. The validation of the discovered turn constraints is done by visualizing each of the detected constraints and using an external mapping service and cross-validating the results. The accuracy of the verified turning restrictions in the three cities and in the two considered boundaries ranges between 57% and 70%.

3.4. Hybrid-based Categorization

Hybrid-based approaches include methods that use a mixture of static and dynamic features. Saremi and Abdelzaher (2015) is the first to initiate such a model, where in addition to the map-based information extracted from OSM, in the availability of dynamic crowd-sourced information (GPS traces), the classification model incorporates features extracted from trajectories, such as the traverse speed, the number of stops and the stopping duration of the last time interval that the vehicle has stopped and idling. 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. Moreover, in cross-city learning transferability settings, classification remains as high as 96%.

Liao et al. (2021) described a traffic light detection (binary classification problem) and impact assessment framework, which can detect the presence of traffic lights and estimate the influence range of traffic lights (vehicle wait queuing in space and time) using speed time series extracted from GPS trajectories and intersection-related features, such as intersection type (connecting arterials, connecting minor roads, connecting arterials and minor roads), road type (according to two speed limits) and traffic flow information. The Dense module is combined with a Long Short-Term Memory neural network (DLSTM) in the proposed framework, which handles discrete and sequential features separately, achieving an AUC value below the ROC curve of 0.95. No information on the size of the dataset is given, and the groundtruth maps is said to have been conducted manually.

Cheng et al. (2022) launches an automatic way to identify three categories of traffic regulators (traffic lights, priority signs and uncontrolled intersections) based on a Conditional Variational Autoencoder (CVAE), utilizing GPS data collected from vehicles and satellite images retrieved from Google Maps. A Long Short-Term Memory network is applied to extract the motion dynamics in a GPS sequence crossing an intersection. In addition, a Convolutional Neural Network (CNN) is build to extract local grid-based image information associated with each step of GPS locations. A self-attention mechanism is adopted to extract the spatial and temporal features in both GPS and grid sequences. The extracted temporal and spatial features are then combined for the classification task of target labels. The methodology is tested on a dataset collected from the city of Hanover consisting of 3,538 intersection arms and 1,204 trajectories. The groundtruth map of traffic rules was generated from on site observation. The dataset is not available as an open dataset to reproduce the results or to use it for testing similar methods. No tests were conducted on the transferability of learning between cities or the effect of the number of trajectories on classification performance. Compared to a Random Forest model and an Encoder-Decoder model, the proposed model achieved better results, with accuracy and F1 score found 0.90.

3.5. Discussion

A common element of the methodologies examined is that classification is done at the intersection arm level rather than the entire intersection. This is motivated from the fact that there are intersections where not all arms are regulated with the same type of regulator and therefore it makes sense to classify each intersection arm separately. If there are large datasets available where there are enough intersection examples representing different regulation mixtures in intersection level, it might also be possible to classify at the intersection level. In such a case, a three-way-all-stop controlled intersection could be labeled as 1, a three-way intersection with one stop and two priority controlled arms could be labeled differently, etc. Because the classification is done at the intersection arm level, the classification features in all but one study represent information exclusively relevant to that arm. Only in the work of Saremi and Abdelzaher (2015) and for the map-based model, the classification features for an intersection arm contain information also from neighbouring arms of the same junction. To the best of our knowledge, no other methodology proposed to date, for example from the dynamic category, 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 for each arm itself.

Even using information from neighbouring arms, the classification is still done at the intersection arm level. Motivated by the single fundamental rule that Saremi and Abdelzaher (2015) used to correct possible misclassifications in traffic light controlled intersections (see Section 3.2.1) and the labeling rule that Hu et al. (2015) use to label the arms of T-shaped intersections that are excluded from classification due to the elimination of turn trajectories (see Section 3.3.1), we believe that checks of predicted labels at a step after classification could trigger a mechanism for recovering incorrect predicted labels based on domain knowledge rules that preserve label consistency at the intersection level. In this way, possible misclassifications of intersection arms could be corrected.

The effect of turning trajectories on classification performance has been examined only in one study (Golze et al., 2020), while the work of Hu et al. (2015) only mentions that it ignored turning trajectories from the feature computation. All other studies make no reference on this aspect of the problem.

Moreover, from Table 3.2 we see that six classes of traffic regulators (TS, SS, UN, TR, YS, PS) were identified as detectable by crowdsensing means from the thirteen articles reviewed. A first observation on this is that the most frequently detectable classes are TS and SS. A second observation is that all of these published studies use different datasets, and apart from an open small dataset (10 trajectories) reported and used by Munoz-Organero et al. (2018), none of the studies made the dataset used publicly available to the research community so that other studies can use it as a benchmark. Without such a dataset as a reference, we observe that each study tests a different methodology on different target labels, with features computed from different trajectory densities (e.g. 1,204 trajectories used in Cheng et al. (2022) and 55 trajectories in Munoz-Organero et al. (2018)), and using different training/testing settings (e.g., 50 intersections in the dataset used by Pribe and Rogers (1999) and 463 by Hu et al. (2015)). For example, can a method that identifies TS and SS with high accuracy, predict equally well the three classes of TS, PS and UN regulators? That is, if we apply the same methodology to a different regulator context, will it perform similarly? This observation highlights therefore the need, a method to be tested on various datasets each containing different regulator labels, so that the generalisation ability of the classifier to be assessed under broader label settings (label categories). Similarly, it is important to test a method on datasets with different trajectory density settings, as the classification features in most methodologies are extracted as the aggregation value of the mean, minimum, etc. of physical features (e.g., number of detected episodes). Moreover, the description of the datasets or classification reports often do not include important information about the number of data samples per class and, therefore, one cannot fully evaluate the classification performance of the proposed methodologies or compare the performance between different recognition approaches.

The method for obtaining the ground truth map in 10 out the 13 studies is based solely on on-site (direct) observation. One study uses street-level images from Google Street View (Saremi and Abdelzaher, 2015), another uses on-site observation and sources from an official transport agency (Pribe and Rogers, 1999), and a third study (Efentakis et al., 2017) uses information from satellite images and on-site examination. Given that on-site observation is also included in these multiple sources, the total percentage of methods that uses the manual method of acquiring ground truth maps is further increased to over 92%. Since the learning process requires the regulator labels (training), the groundtruth map of the regulators is an integral part of the data needed by the algorithms. When its acquisition requires a manual process (on-site observation), this imposes constraints related to time and cost on recognition methods that intend to automate the *overall* process of identifying traffic regulations.

Thus, if by crowdsensing or using platforms that provide such data, a lot of trajectory data can be readily available to address the classification task, obtaining the ground truth map is still necessary for learning purposes. Today, many GPS datasets are available from various open source platforms or institutions or competitions, but cannot be used in the context of traffic regulator recognition unless the ground-truth map is also provided. Therefore, this finding highlights the need for open datasets that researchers can easily access, so that they do not have to come across the time-consuming manual work that a groundtruth map entails.

Moreover, on the one hand the importance of regulator labels in the context of supervised classification and on the other hand the time-consuming and costly task of constructing groundtruth maps, motivate the investigation of unsupervised and semi-supervised methodologies that could be used in the context of this problem and so far have only been explored by one study (Hu et al., 2015).

Another important conclusion is that the transferability of learning between cities remains an unexplored aspect of the problem, as only one study has conducted experiments on this (Saremi and Abdelzaher, 2015). Similarly, an unexplored issue is the number of trajectories required for sufficient classification: 5, 10, 20, or how many trajectories are required for a classifier to learn to discriminate between different regulators?

Additionally, more conclusions could be drawn if classification performance was assessed across all studies with the same metrics, or even better, if detailed classification reports were provided, including the number of data instances per regulatory class used to build the classification models. It seems that hybrid methods (Class.Perfom, and Class. Method columns in Table 3.2), such as that of Saremi and Abdelzaher (2015), 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 (Saremi and Abdelzaher, 2015; Cheng et al., 2022), it may be an interesting methodological direction to explore further.

A general observation already outlined in Section 1.2 is that crowd-sourcing traffic regulators for map-enrichment is a less explored topic compared to other automatically generated map elements (e.g., street geometry, intersection locations). This is also reflected by the number of articles excluded in the screening process for the review of published work in the field, as well as the small number of articles that make up the list under review of this chapter. The recent interest on this subject, if taken into account the long period of inactivity since 1999 when the first relevant article was published, could be explained by the practical need to enhance maps with this information. Therefore, considering the limitations of existing methodologies, as described in this section, new research directions can be made for the topic explored in this thesis.

3.6. Knowledge Gap

We identify the following major research directions, that are either unexplored or under-explored so far by existing works, and therefore this thesis aims at:

1. Proposing and testing a new traffic regulator recognition methodology and evaluating it in different dataset settings, i.e., different cities, regulators, trajectory densities and dataset sizes.
    - Since, from the literature review it was identified that hybrid methods seem to perform better, the proposed model will examine further this finding.

   - The proposed methodology will be tested under different feature settings, i.e., including information from context arms and using exclusively information from one intersection arm.
   - The methodology will propose an additional consistency check of the predicted labels at an intersection level, at a post-processing step, recovering incorrect predicted regulator labels when possible.
   - The proposed methodology will examine the influence of sampling rate of GPS tracks on the classification performance.

2. Investigating the classification performance of the proposed method under different trajectory settings.
   - We will examine whether there is a certain number of trajectories per intersection arm that leads to optimal classification performance.
   - We will examine the effect (if any) of turning trajectories in the classification performance.
3. Investigating the classification performance of the methodology under sparsely labeled data and streaming data.
   - The classification performance of the proposed methodology will be examined under: 1) no available labeled data, and 2) the availability of various amounts of labeled data (12 labeled data instances, 24, etc.). Machine learning methods such as clustering, semi-supervised learning techniques such as self-learning, as well as active learning, will be used for the above purpose.
   - The proposed classification method will be assessed under learning transferabilty settings (train a classifier on city A and predict regulators on city B).
   - The proposed methodology will be applied under the framework of incremental learning. Instead of processing all data at one time for building a learning model, data instances will be processed one at a time and the learning model will be updated on an incremental way (training with a single data instance).

3.7. Acknowledgements

The observations, discussions and conclusions of Sections 3.5 and 3.6 have been partly published in the ISPRS International Journal of Geo-Information under the title *Traffic Regulator Detection and Identification from Crowdsourced Data-A Systematic Literature Review* (Zourlidou and Sester, 2019).

The review of some articles mentioned in this chapter has been also published in the ISPRS International Journal of Geo-Information under the title *Recognition of Intersection Traffic Regulations From Crowdsourced Data* (Zourlidou et al., 2023).

The thesis author is the principal investigator and author of both articles, as well as the creator of all figures.

4. Traffic Regulation Recognition (TRR) from GPS Data

4.1. Introduction

This chapter discusses four different TRR methods that according to the proposed TRR taxonomy (Section 3.1) fall into the categories *static*, *dynamic* and *hybrid*. The first method is called *static* (Section 4.3.2) and uses features derived from OSM. The second method Section 4.3.3) is a dynamic TRR approach, called *c-dynamic* (c- for compact), which uses features describing the speed of vehicles on approaching the intersection, as well as their movement behaviour as captured by four movement patterns: (a) unhindered crossing of the intersection, (b) deceleration without stopping, (c) stopping once and (d) stopping more than once before crossing the intersection. The third method (Section 4.3.4) is called *dynamic* and is an extension of the c-dynamic method using all the features of the latter and other features related to stopping and deceleration episodes. The fourth method (Section 4.3.5) is called *hybrid* and uses a combination of the features of the static and dynamic models together. For the four TRR methods, two supervised algorithms are considered for building the learning models, the Random Forest (RF) and the Gradient Boost (GB), using three different datasets from the city of Hanover (DE), Champaign (US) and Chicago (US), which are described in Section 4.2. The predictive performance of the trained classifiers is given in Section 4.4 and discussed in Section 4.5.

4.2. Datasets

This section explains the data requirements for dealing with the TRR problem from crowdsourced data, and the constraints arising from these requirements (Section 4.2.1). Section 4.2.2 describes the data used in this thesis. The construction process of the groundtruth map of intersection regulations is explained in Section 4.2.3.

4.2.1. Dataset Requirements and Limitations

As the proposed four TRR methods are based on supervised classification, both GPS traces (for feature computation) and regulators (as labels) of the intersection arms are required. Labeling, as already discussed (Section 3.5), is a time-consuming process and open trajectory datasets cannot be used unless information on the intersection regulations is also available to be used as target labels for training/testing purposes. With regard to the trajectory dataset, an important requirement is that GPS samples must be recorded at a high frequency, for example, 1 sample per second. This requirement stems from the fact that stopping and deceleration episodes (classification features) are estimated from the GPS tracks and therefore, if the sampling frequency is low (e.g., 1 sample every 15 s), short duration episodes, such as those of stopping and deceleration, would occur between two samples and could not be detected. Some widely used trajectory datasets such as the Athens and Berlin datasets (Ahmed et al., 2015) have sampling intervals 30.14 s and 41.98 s, respectively. Therefore, this requirement further limits the possible open trajectory datasets that could be used for TRR, due to the required sampling rate. Up to the time of the implementation of this thesis, no open dataset with a sampling rate above 0.28 Hz (≈1 sample every 3.61 s) was found, except for the Chicago dataset (Ahmed et al., 2015). Thus, having to address these challenges of datasets, the thesis author was able to access three appropriate datasets in total: one recorded by the thesis author and now available as open dataset (Zourlidou et al., 2022a,c), one shared by the author of the journal article Hu et al. (2015) at the request of the thesis author, and one open trajectory dataset (Ahmed et al., 2015) for which the groundtruth map had to be manually conducted by the thesis author and is now available as an open dataset, too (Zourlidou et al., 2022b).

4.2.2. Datasets for Testing the Proposed Methods

Table 4.1 gives a description of the three datasets. The datasets contain two combinations of rules, with the Champaign and Chicago datasets containing the same set of rules (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 one regulator (UN, Priority Sign (PS), TS). Although the Hanover dataset contains Yield Sign (YS) controlled arms, most of them are sparsely sampled (few trajectories cross them) and only a few of those sampled from more than 3 trajectories could in principle be used for training/testing. For this reason YS was excluded from the analysis.

One regulation is considered per intersection arm, which means that a 3-way intersection has three regulations and a 4-way intersection has four regulations. However, not all intersection arms are sampled from the trajectories, i.e., an intersection may be sampled from only one of its roadways (approaches), while another may be sampled from all of its roadways. Therefore, depending on the types of intersections (3-way, 4-way) and the availability of trajectory samples that cross the intersection arms, the total number of regulations per dataset varies accordingly. The *Hanover* dataset consists of 1,204 tracks with a total length of 6,498 km (average: 5.39 km and standard deviation 4.14 km) obtained from a vehicle that the thesis author used for her daily commuting covering an area of 11.6 km x 16.9 km, with an average sampling rate of 1.7 s and average speed 30.28 kmh. The tracks cross 1,064 intersections with 3,538 in total intersection arms. The *Champaign* dataset consists of 2,022 tracks with a total length of 6,230 km (average: 2.83 km and standard deviation 3.98 km) obtained from vehicles driven by 35 volunteers covering an area of 12.2 km x 13 km, with an average sampling rate of 1 s and average speed 35.1 kmh. The tracks cross 713 intersections with 2,501 in total intersection arms. The *Chicago* dataset consists of 889 tracks with a total length of 2,869 km (average: 3.22 km and standard deviation 897.28 m)) obtained from university shuttle buses covering an area of 3.9 km x 2.5 km, with an average sampling rate of 3.61 s and average speed 32.86 kmh. The tracks cross 156 intersections with 568 in total intersection arms. All data are naturalistic[1] in the sense that drivers were not given external instructions on how to drive. Figure 4.1 illustrates the three datasets.

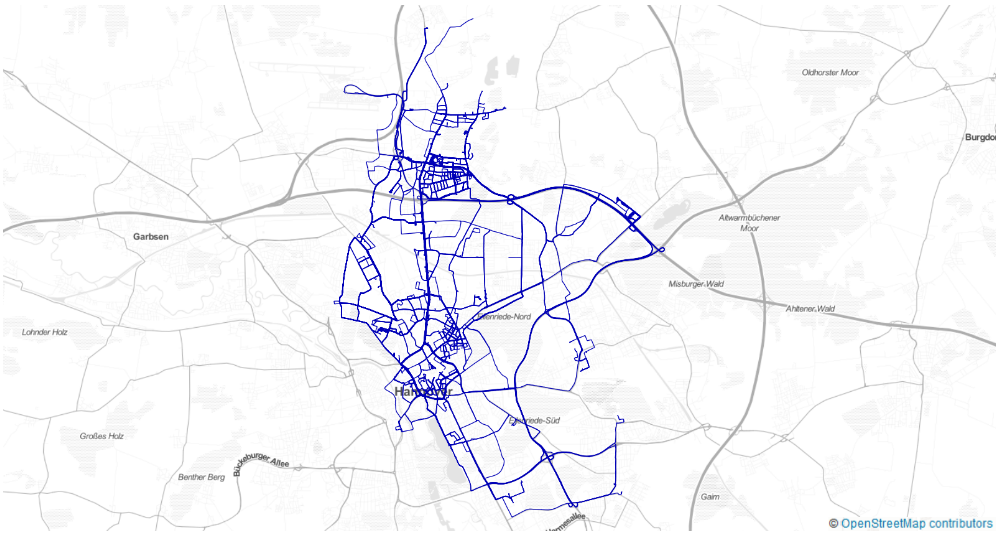
Table 4.1.: Datasets used for testing the proposed methods.

City	**Junc.**	**Rules**	**Traj.**	**Avg. Speed(km/h)**	**Avg.Traj. Length(km)**	**Sampl. Rate(s)**	**Rules** *
Hanover	1,064	3,538	1,204	30.28	5.39	1.7	TS, PS, UN
Champaign	713	2,501	2,202	35.1	2.83	1	TS, SS, UN
Chicago	156	568	889	32.86	3.22	3.61	TS, SS, UN

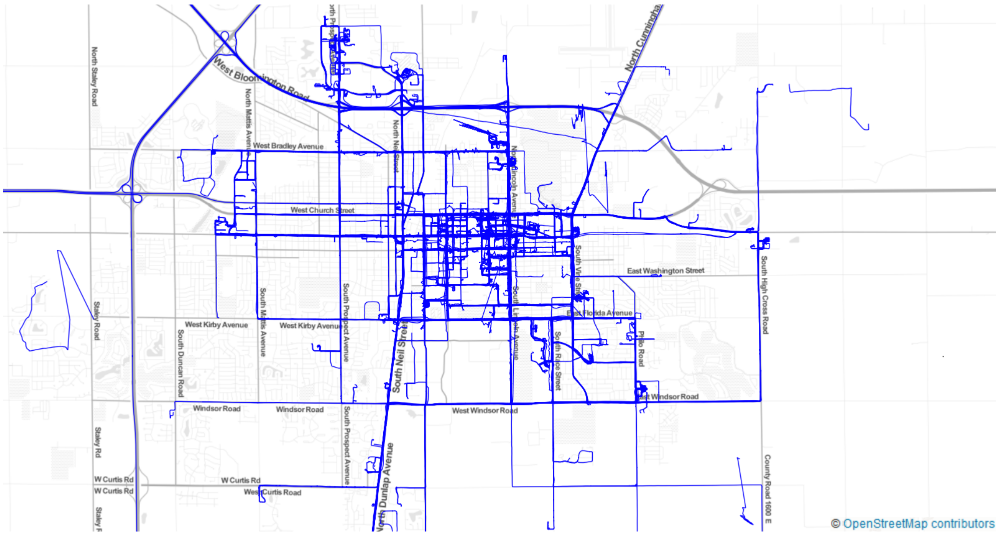
*TS: Traffic Signal, SS: Stop Sign, UN: Uncontrolled, PS: Priority Sign.

---

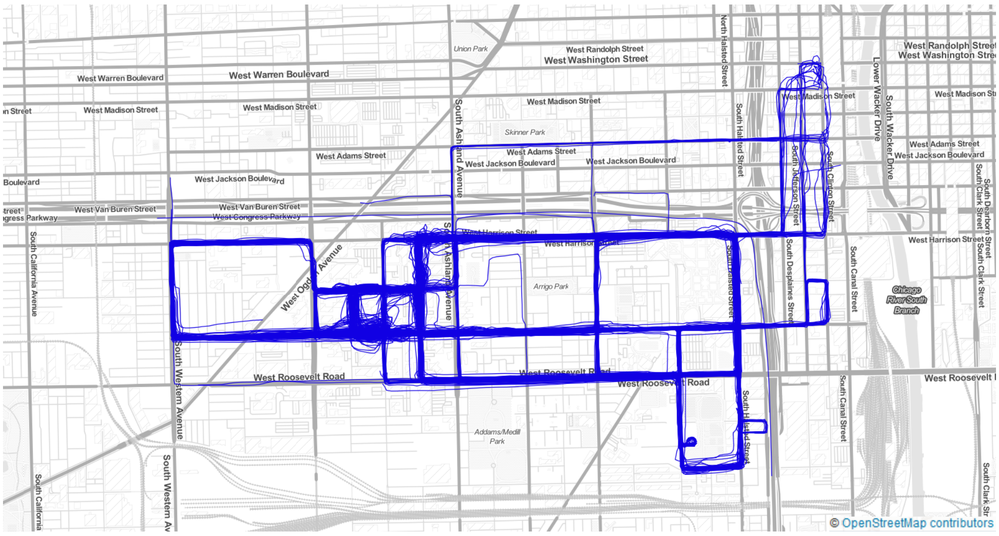
[1] Naturalistic data can be defined as data that make up records of human activities that are neither elicited by nor affected by the actions of researchers Given (2008).



(a) Hanover.



(b) Champaign.

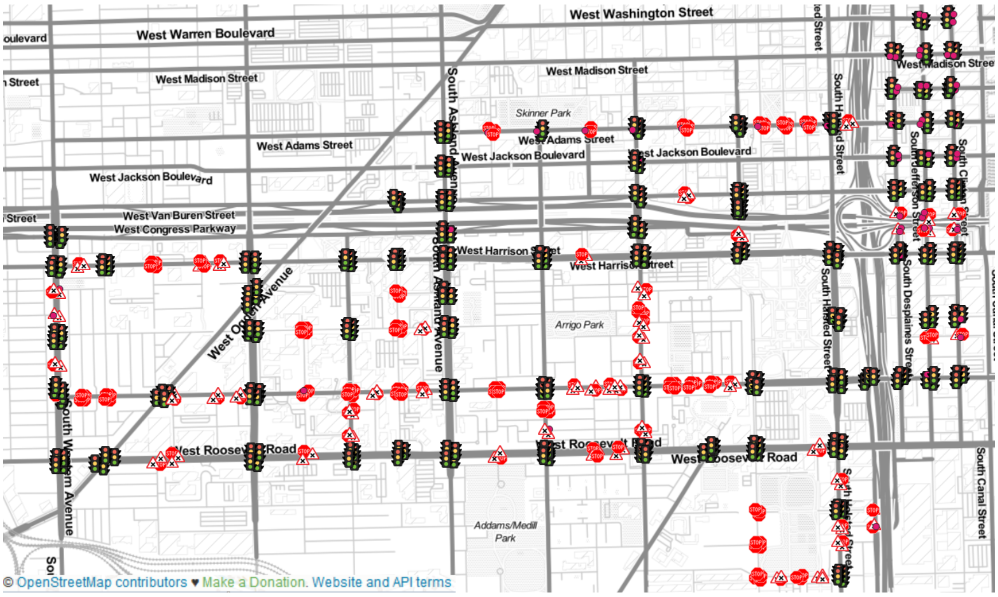


(c) Chicago.

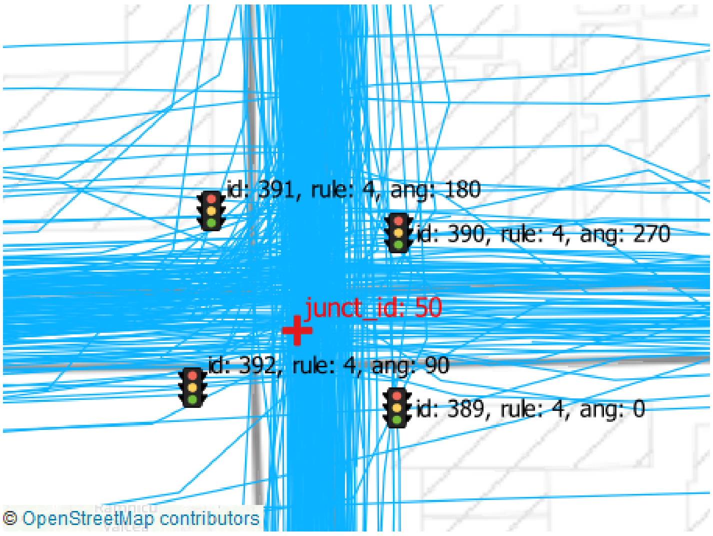
Figure 4.1.: The three datasets used in this study.

4.2.3. Groundtruth Map Construction

The purpose of a groundtruth map of intersection regulations is to relate each intersection arm to the type of regulation it is controlled by. Such a map is illustrated in Figure 4.2a and an example of the recorded information per intersection arm (arm_id, rule_label, arm_angle, junction_id) is shown in Figure 4.2b. To construct the groundtruth map of the Hanover dataset, the mobile phone application *mapillary* (Mapillary, 2022) was used to capture street images. A mobile phone was placed in the front window of the car, as shown in Figure 4.3a, and then, running the app, geotagged images were recorded while driving (Figure 4.3a). Each mapillary logging session results in a sequence of geotagged images, which can either be viewed from the application's *Upload* tab (Figure 4.3c) or the images can be uploaded to a Geographic Information System (GIS) (we used QGIS) and then viewed using the provided event browser tool from the GIS (Figure 4.3b).



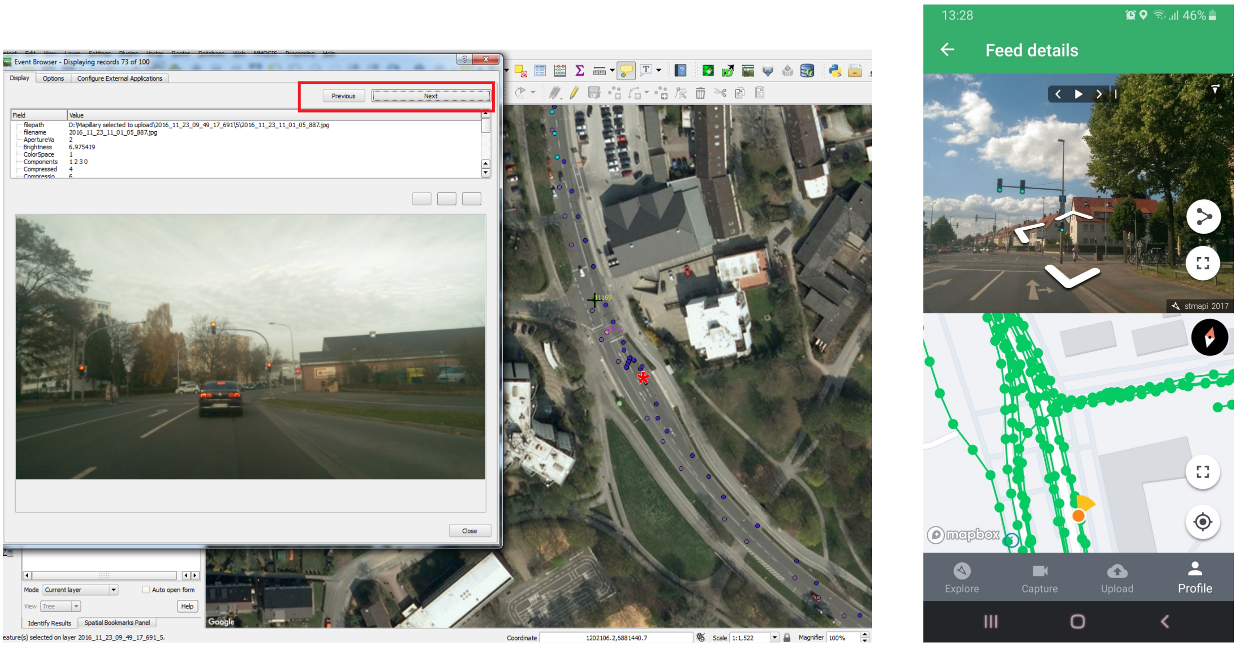
(a) The groundtruth map of the Chicago dataset.


id: 391, rule: 4, ang: 180
id: 390, rule: 4, ang: 270
junct_id: 50
id: 392, rule: 4, ang: 90
id: 389, rule: 4, ang: 0
© OpenStreetMap contributors

(b) The recorded information on a groundtruth map.

Figure 4.2.: Examples of groundtruth maps.

(a) Capturing street level photos with mapillary App.



Field	Value
filepath	D:\Mapillary selected to upload\2016_11_23_09_49_17_691\5\2016_11_23_11_01_05_887.jpg
filename	2016_11_23_11_01_05_887.jpg
ApertureVa	2
Brightness	6.975419
ColorSpace	1
Components	1 2 3 0
Compressed	4
Compressio	6

(b) Examining mapillary captured photos in Qgis for labeling intersection arms.

(c) Examining mapillary captured photos in mapillary application.

Figure 4.3.: Tools for constructing groundtruth maps of intersection regulations.

We used the latter method to examine the image sequences one by one (previous and next image buttons shown in Figure 4.3b), while simultaneously examining the position of the captured image on the map (red asterisk in the largest window of Figure 4.3b). The traffic rule (label) of the inspected intersection arm was then manually recorded (at a separate layer in QGIS) and given a unique identification number, and all these features were associated with the relevant intersection (intersection identification number) to which the corresponding intersection arm belongs (Figure 4.2b).

For the Chicago groundtruth map, mapillary images uploaded by other users were examined and the groundtruth map was constructed using the QGIS software again. Some labels, when possible, were verified by examining additional Google satellite imagery. The groundtruth map from the Champaign dataset was provided to the thesis author as part of the TRR dataset shared for research purposes from the first author of the article (Hu et al., 2015), after request of this thesis author. However, some intersections were not represented in the groundtruth map even though they were crossed by trajectories, and we had to add their labels using the same method as in the other two datasets.

4.3. Methodology

This section describes the clustering algorithm for detecting short-term significant movement episodes, such as stopping and decelerating episodes (Section 4.3.1). These episodes are then used as classification features per se or further processed to compute other classification features used by the four proposed TRR methods (Section 4.3.2, 4.3.3, 4.3.4, 4.3.5).

4.3.1. Detection of Stop and Deceleration Episodes

The Clustering Based Detection of Stop and Deceleration Episodes of Trajectories (CB-SDoT) algorithm (see Algorithm 10) is a modification of the CB-SMoT (Palma et al., 2008) algorithm (Section 2.2.7) for detecting significant episodes of short duration. Usually significant locations as discussed in Section 2.2.6 are identified as those places that the moving objects stay a significant amount of time. In the context of the work examined in this thesis, the hypothesis is that the time spent at the intersections can be distinctive compared to non intersection locations but the time intervals are much smaller than those locations recognised by algorithms, such as the CB-SMoT, (touristic locations for example). Moreover, although in the context of other stop-and-move detecting algorithms the more time a moving object spends at a certain location, the more significant the location is, such a concept is neither valid nor desirable under the TRR methodology that this thesis examines. Stopping episodes that have a large duration should be considered as outliers as they may indicate traffic jams or parking alongside the road and the source that causes them in general is not a traffic regulation. For this reason, CB-SDoT identifies clusters of points (Figures 4.4 and 4.5) that within a certain distance Eps remain at least $minTime$ and (unlike CB-SMoT) no more than $maxTime$. The $maxTime$ parameter limits longer stops to be distinguished as stopping episodes. In total, CB-SDoT in addition to the distance between points (Eps), it takes into account the temporal distances between them to determine the clustering criteria. The values of the parameters were defined experimentally. For stopping episodes: Eps=10 m, $minTime$=4 s, $maxTime$=600 s, and for deceleration episodes: Eps=10 m, $minTime$=2.4 s, $maxTime$=3.9 s.

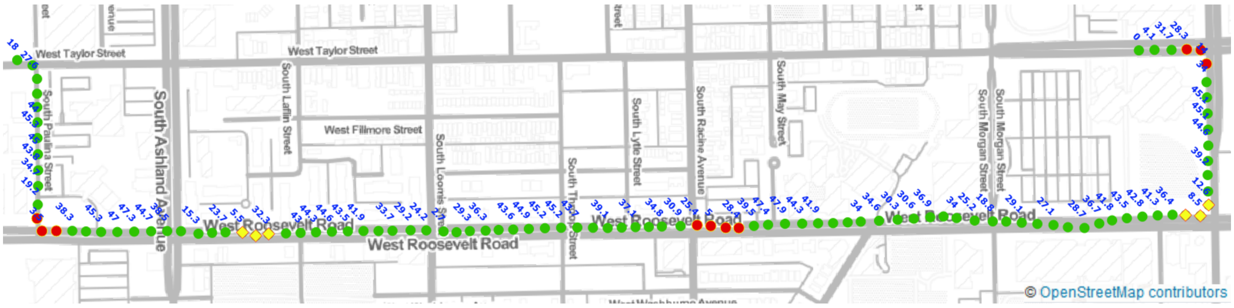


Figure 4.4.: Stopping (red) and deceleration (yellow) episodes detected in one trip (from east to west). The numbers in blue indicate the vehicle's speed in kmh.

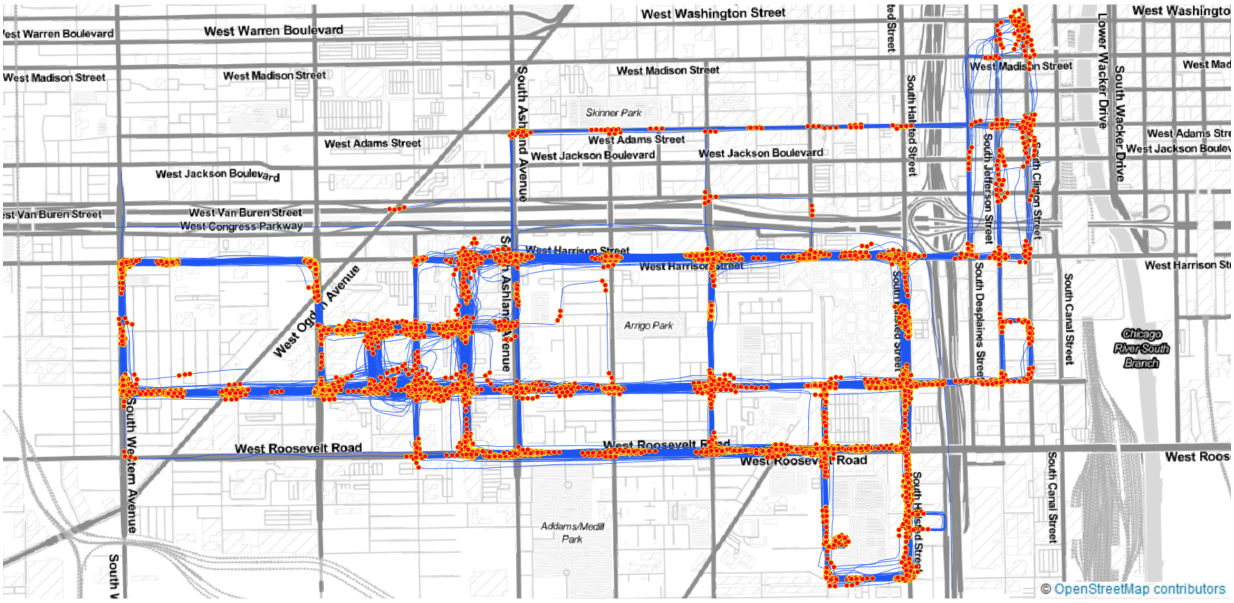


Figure 4.5.: Stopping episodes (in red) detected in the Chicago dataset.

Algorithm 10: The CB-SDoT algorithm.

```
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
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 for
            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 if
    end for
end for
```

The concepts *core point*, *linear neighbourhood* and *neighbouring points* (Algorithm 10) refer to the same ones originally defined in Ester et al. (1996) and Palma et al. (2008) and already given in 2.2.7 and 2.3.3.2. Each detected cluster is a temporally ordered sequence of points. For each cluster, a point is identified as the *representative* point of the cluster. Such a point could be a core point of the cluster, or the first point in the time series of the cluster (indicating the start of the event), or the last point (indicating the end of the event), or the middle point, or the point closest to the centre of the intersection, or the one with the lowest speed and closest to the centre of the intersection. The Algorithm 10 considers such a point to be the last point in the time series of points in the detected cluster that has the lowest speed. The reason for defining such a point is that additional classification features are extracted from the stopping and deceleration episodes, which are related to the distance of the episodes from the centre of the intersection. One such example is the distance of the last stopping episode from the center of the intersection, (see dynamic-model at Section 4.3.4). Measuring such distances requires a point that represents the cluster (to measure from that point to the point that represents the intersection center), and the representative point serves exactly this role.

4.3.2. The Static Approach

The static method uses five features per intersection arm, *all* extracted from OSM. Originally these features were proposed by Saremi and Abdelzaher (2015) for the map-based TRR model, but there each intersection arm is described by a feature vector that combines features from all intersection arms belonging to the same intersection, i.e., an intersection arm of a 4-way intersection is defined by 4x5=20 features. Here we propose a simplified TRR model of the original, where each intersection arm is described by only five features (depicted in Figure 4.6):

1. the **end-to-end distance** of the road that the intersection arm belongs to (red arrow in Figure 4.6). The length of a road is indicative of its importance in the road network. The same rationale applies also to the other distance-based features (points 2, 3).
2. the **semi-distance** of an intersection arm is the distance from the center of the intersection to the center of the most distant intersection that the intersection arm is connected to (green arrow in Figure 4.6).
3. the **closest distance** of an intersection arm is the distance from the center of the intersection that the arm belongs to, to the center of the nearest intersection that the arm is connected to (light blue arrow in Figure 4.6).
4. the **maximum speed** (speed limit) of an intersection approach is the maximum allowed speed along it. Intersections controlled by 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 street).

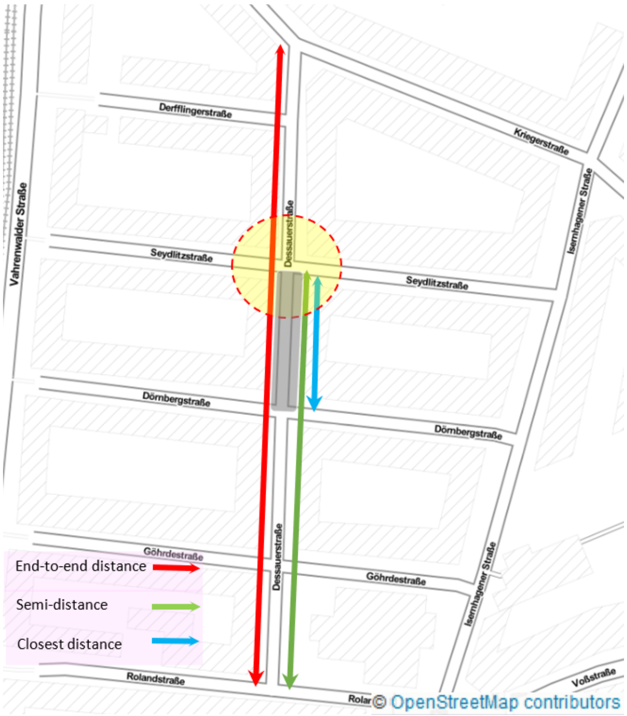


Figure 4.6.: Illustration of the distance-related features (end-to-end distance, semi-distance and closest distance) of the static method that represent the north-south intersection arm (indicated in grey color) of a four-way intersection (in yellow).

4.3.3. The c-Dynamic Approach

This proposed TRR 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 episodes, as well as their non-observation, that is no stop and no deceleration episodes (four pattern blocks). For example, a movement pattern can be a free crossing of an intersection where no stop or deceleration episode is observed. Another pattern can be stopping only one time before crossing the intersection. Numerous patterns can be defined by combining these patterns blocks. Then each regulated intersection arm can be described from the movement patterns observed at its location, by simply summarizing the patterns (each described by stop/deceleration episodes) of all the trajectories that cross that intersection 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 the TRR problem, we define four different movement patterns, depicted in Figure 4.7, 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 (no deceleration or stopping episodes 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.

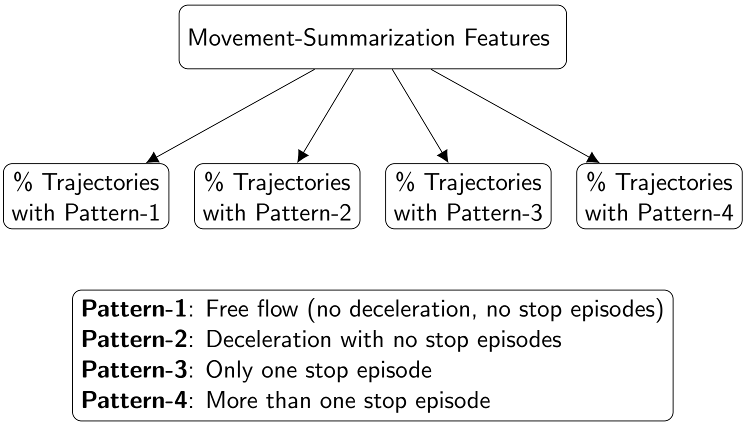

Movement-Summarization Features
% Trajectories with Pattern-1
% Trajectories with Pattern-2
% Trajectories with Pattern-3
% Trajectories with Pattern-4
Pattern-1: Free flow (no deceleration, no stop episodes)
Pattern-2: Deceleration with no stop episodes
Pattern-3: Only one stop episode
Pattern-4: More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

Figure 4.7.: The four movement patterns that describe a vehicle's crossing of an intersection arm.

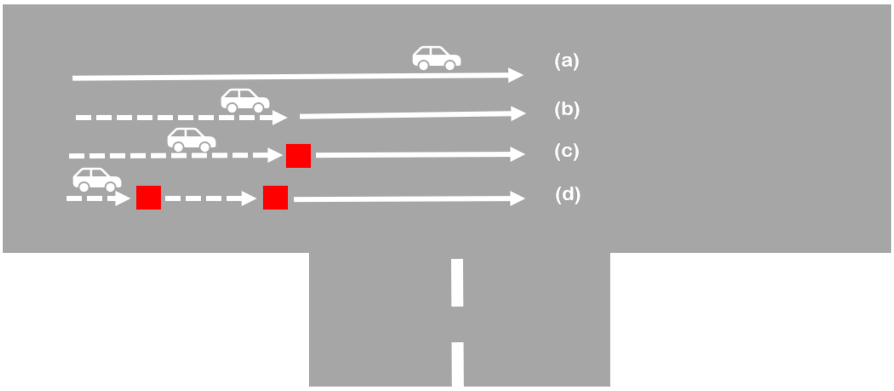
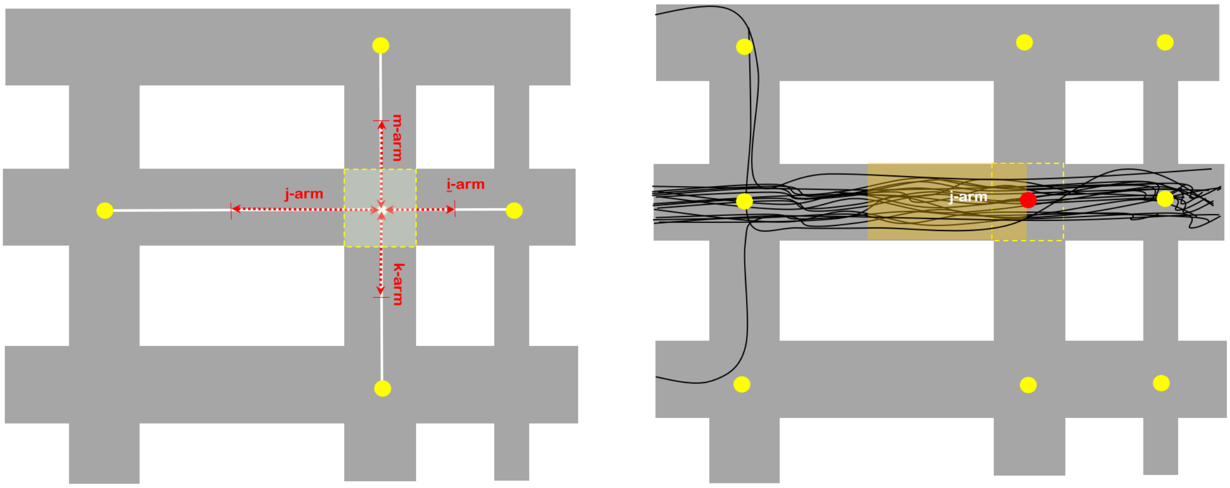


Figure 4.8.: The four movement patterns that describe a vehicle's crossing of a junction: (a) unhindered crossing, (b) deceleration (dotted line) without stopping, (c) stop once (red square), (d) stop more than once (here two stop events are depicted with two red squares).

Schematically, this idea is illustrated in Figure 4.8. Such a mixture of motion patterns has been used in Tang et al. (2016), 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. At a yield sign, we observed patterns 2 and 3 to have higher values compared to patterns 1 and 4.

Along with these four patterns computed per intersection arm, that are used as classification features, we add in the feature vector additionally the 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). Regarding the implementation of this idea, all steps, from feature extraction to intersection classification, are expressed in Algorithm 11. First, stopping and deceleration episodes in trajectories are detected using the Algorithm CB-SDoT. Then, for each intersection arm of the dataset, the classification features are computed from the trajectories crossing it. As illustrated in Figure 4.9, for an intersection that has four arms, i_arm, m_arm, j_arm, k_arm, for each arm and for each trajectory that crosses it, the stopping and deceleration episodes within *half* the distance connecting the intersection to the previous visited intersection (white arrows) are computed. Other existing TRR methods, compute features within a buffer of a given size. For example, in Golze et al. (2020), stopping episodes are calculated within 100 m of the intersection center. However, such feature calculations within fixed-size distances can be problematic in areas where one intersection from another is closer than this distance. Therefore, the proposed approach avoids such a problem. Then, according to the stop/deceleration episodes detected on each trajectory crossing an intersection arm, the movement behavior of the trajectory is categorized into one of four movement patterns. The percentages of trajectories that follow each pattern per intersection arm of the dataset are then calculated.



(a) *(b)*

Figure 4.9.: Each intersection consists of intersection arms that connect it to nearby intersections. Classification features are calculated per arm, within half the distance of the road segment connecting the current arm to the previous arm visited by the trajectory (red dashed arrows in (a)). For each trajectory in (b) that crosses the intersection arm j-arm from west to east, stopping and deceleration episodes are detected within the orange indicated area along the j-arm in (b).

Algorithm 11: TRR from GPS tracks (c-dynamic approach).

```
Data: GPS tracks, ground truth map
Result: Predict the traffic regulators by which intersection arms are controlled
while not all trajectories have been processed do
    Find all stopping and deceleration episodes in the trajectory
    Add stopping and deceleration episodes in DB tables $StopTB$ and $DecTB$
end while
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 behaviour (num. of stop/deceleration events) of the trajectory
          $Trj$ to one of the four patterns
        Estimate the average crossing speed
    end for
    $p_1,p_2,p_3,p_4 \leftarrow$ Compute the % of the four patterns for junction arm $i$
    $s_1,s_2,s_3,s_4,s_5,s_6 \leftarrow$ Compute the 10th, 20th, 40th, 60th, 80th and 95th average speed
      percentiles for junction arm $i$
    Add feature vector $p_1,p_2,p_3,p_4,s_1,s_2,s_3,s_4,s_5,s_6$ to DB table $FeaturesTB$
end for
Training and testing a classifier with data from $FeaturesTB$
Print classification report
```

4.3.4. The Dynamic Approach

The dynamic approach can be considered as an extension of the c-dynamic model, using, in addition to the ten features used in the latter model, some statistical features (average, variance, minimum and maximum values) derived from the stopping and deceleration episodes and from the estimated vehicle speed. Compared to existing TRR methods, for example, those of Hu et al. (2015); Golze et al. (2020); Saremi and Abdelzaher (2015); Carisi et al. (2011), this approach has a richer feature vector (86 features in total) that includes, in addition to stopping features, deceleration- and speed-related features, and the calculation of the movement episodes is computed within a non-fixed distance from the intersection center (see Section 4.3.3). All classification features are listed in Table 4.2.

Table 4.2.: Overview of the classification features derived for the dynamic TRR approach.

	#	Physical Feature*	Statistical Features **			
Stopping episodes	32	Number of stop epis.	avg	var	min	max
		Duration of last stop epis.	avg	var	min	max
		Duration of all stop epis.	avg	var	min	max
		Mean Duration of all stop epis.	avg	var	min	max
		Median Duration of all stop epis.	avg	var	min	max
		Distance of last stop epis.	avg	var	min	max
		Mean Distance of all stop epis.	avg	var	min	max
		Median Distance of all stop epis.	avg	var	min	max
Decel. episodes	32	Number of decel. epis.	avg	var	min	max
		Duration of last decel. epis.	avg	var	min	max
		Duration of all decel. epis.	avg	var	min	max
		Mean Duration of all decel. epis.	avg	var	min	max
		Median Duration of all decel. epis.	avg	var	min	max
		Distance of last decel. epis.	avg	var	min	max
		Mean Distance of all decel. epis.	avg	var	min	max
		Median Distance of all decel. epis.	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. Patterns	4		Traj. % with no stop/decel. episodes			
			Traj. % with decel. episodes			
			Traj. % with one stop episode			
			Traj. % with more than one stop			
Sum	86					

* Derived per trajectory, ** Derived from the physical features per intersection arm.

4.3.5. The Hybrid Approach

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

4.3.6. Implementation and Classification Settings

Two tree classifiers are used for the classification of the intersection arms: the classifier RF and the classifier GB. The *XGBoost* library (Chen and Guestrin, 2016) and the *sklearn* library (Pedregosa et al., 2011) were used to implement the GB and RF respectively. All programming tasks were implemented in *Python 3.7*. All data (GPS tracks, regulation labels, centers of intersections, OSM map) were stored in a PostgreSQL open source Database (DB) with PostGIS extension[2]. All the data required by the program are retrieved through SQL queries from the DB. For the extraction of the dynamic features, only the GPS tracks (DB table *GPS Tracks*), the center of intersections (Table *Intersections*) and the angles of the intersection arms (Table *Regulations*) are required. As an example of how the data have been stored in DB is the following; a row from Table *Regulation*, represents a traffic regulation which has a unique identification number *id* (primary key of the Table) and controls the traffic of the arm of intersection *inter_id*. The angle of the arm is *arm_angle* (see Figure 4.10 for the reference angle system of intersection arms). The angle attribute is used for "matching" the trajectories to intersection arms so that the dynamic classification features can be extracted accordingly. For the extraction of the static features, the OSM of the area of interest in needed (*osm_point*, *osm_line*).

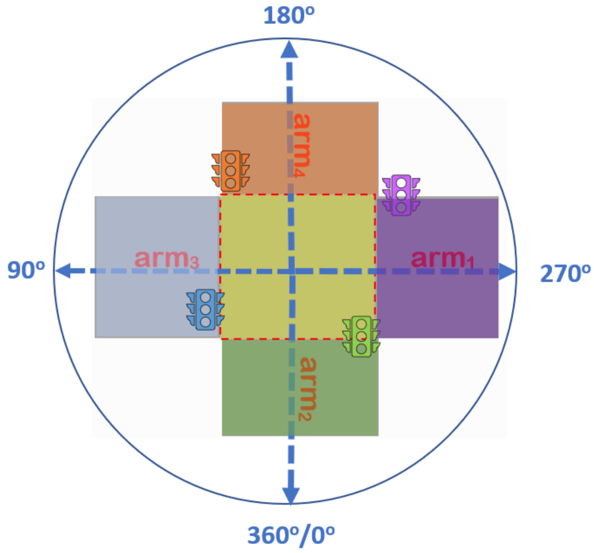


Figure 4.10.: The reference angle system for describing the intersections arms.

As default feature settings are considered the features extracted from straight trajectories (excluding trajectories turning at an intersection). The effect of turning trajectories on classification performance is investigated in Section 5.3.4. In addition, intersection arms crossed by less than five trajectories are excluded from the learning process (the effect of the number of trajectories is investigated in Section 5.3.5).

---

[2] PostGIS is an open source software program that adds support for geographic objects to the PostgreSQL object-relational database.

4.4. Results

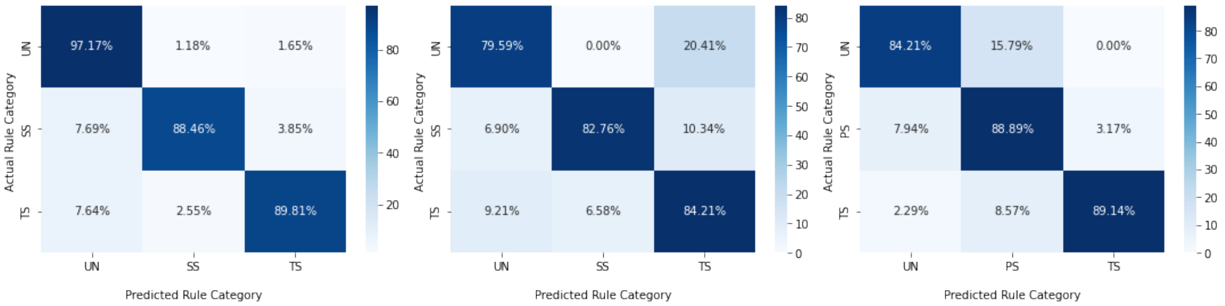
Table 4.3 shows the classification accuracy of the four proposed approaches. The GB classifier performs equally well or better than RF for almost all experiments. Only in the c-dynamic method for the Chicago and Hannover dataset, RF performs slightly better than GB (+10% accuracy (0.1)). In all other cases, GB performs better or equally well as RF. Comparing the four methods to each other, the static model has much lower accuracy than the other models for all datasets. The hybrid approach has the best accuracy compared to the other three methods (0.95 in Champaign, 0.88 in Hanover, and 0.82 in Chicago), and the dynamic approach performs better than c-dynamic but worse than hybrid. In the Champaign and Hanover datasets, the performance of the c-dynamic approach is very close to that of the dynamic approach (-0.1 worse in accuracy), in contrast to the Chicago dataset where the difference between the two approaches is larger (-0.4 in accuracy). Another observation is that in all datasets, the dynamic and hybrid methods perform very similarly: 0.94 versus 0.95 in Champaign (GB), 0.81 versus 0.82 in Chicago (GB) and 0.87 versus 0.88 in Hanover (GB). This means that the dynamic features already have very good predictive ability and the additional static features improve the classification model only slightly (+0.1 in accuracy). However, the hybrid approach under the GB classifier has the best performance on all three datasets. The classification report of the hybrid approach as well as the confusion matrices for the three datasets are given in Table 4.4 and Figure 4.11. From the confusion matrices, one can see that the three classes in each dataset are predicted with similar accuracy and no striking misclassifications “favoring” certain classes are observed. The classes with the highest FPR across the three datasets are UN in Champaign (Figure 4.11d), TS in Chicago (Figure 4.11e) and PS in Hanover (Figure 4.11f).

Table 4.3.: Classification accuracy (Acc) and F1-score (F_1) of the four TRR models. The highlighted number(s) per dataset correspond(s) to the best accuracy achieved in the respective dataset.

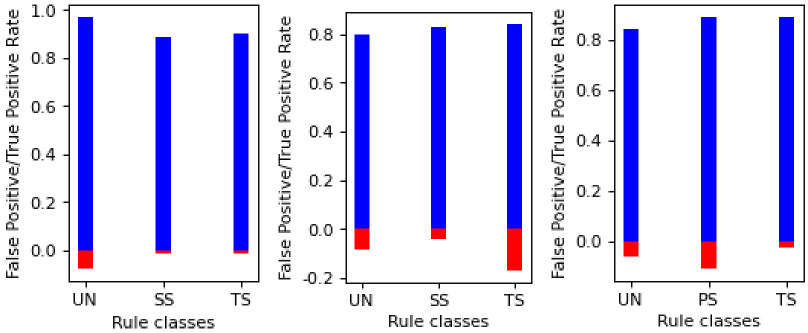
Method	Champaign				Chicago				Hanover			
	RF		GB		RF		GB		RF		GB	
	Acc	F_1	Acc	F_1	Acc	F_1	Acc	F_1	Acc	F_1	Acc	F_1
Static	0.67	0.67	0.69	0.69	0.72	0.71	0.72	0.71	0.61	0.61	0.62	0.62
c-Dynamic	0.93	0.93	0.93	0.93	0.78	0.78	0.77	0.77	0.86	0.86	0.85	0.84
Dynamic	0.94	0.94	0.94	0.94	0.81	0.81	0.81	0.81	0.86	0.87	0.87	0.87
Hybrid	0.94	0.94	0.95	0.95	0.82	0.82	0.82	0.82	0.87	0.87	0.88	0.89

Table 4.4.: Classification results of the hybrid approach (GB).

Dataset	Recall	Precision	F-Measure	Accuracy
Champaign	0.95	0.95	0.95	0.95
Chicago	0.82	0.84	0.82	0.82
Hanover	0.88	0.89	0.89	0.88



(a) Champaign. *(b)* Chicago. *(c)* Hanover.



(d) Champaign. *(e)* Chicago. *(f)* Hanover.

Figure 4.11.: Confusion matrices and false/true positive rates for the three datasets.

4.5. Discussion

The main findings of this chapter are that the hybrid approach under the GB classifier predicts with an accuracy of more than 82% three classes of regulators. Examining the classification results on the three datasets, the following questions are raised:

1. *Why is the performance on the Chicago dataset significantly lower than that on the Champaign dataset?*

   Two possible factors may account for this difference. One has to do with the lower sampling rate at which the Chicago traces are sampled and the other has to do with the size of the Chicago dataset in terms of the number of intersection arms available for training. The fact that the sampling rate in Chicago is only 3.61 s (in Champaign it is 1 s and in Hanover 1.7 s) may affect the episodes detected by the CB-SDoT algorithm and therefore may qualitatively and quantitatively affect the features extracted from the traces. This factor will be addressed in the next Chapter (Section 5.2.2). In terms of the dataset size, the Champaign dataset contains almost 5 times more examples of regulators than the Chicago dataset (2,501 rules vs. 568 rules). This factor will be addressed in a separate chapter later (Chapter 6).

2. *How can we improve the classification performance of the proposed TRR approaches?*

   As already mentioned in the review of existing methods (Sections 3.5 and 3.6), no dynamic approach has considered the use of features from nearby intersection arms of the same junction. Saremi and Abdelzaher (2015) has tested this idea in its map-based model, with very good prediction results. Therefore, it seems a reasonable direction to look for the answer to the above question. In the next chapter, the methodology presented in this chapter, will be tested under an extended feature vector that encompasses information from all arms of the same intersection.

3. *Can classification settings negatively affect classification performance?*

   As explained in Section 4.3.6, only straight trajectories were used in the classification learning process, to avoid a possible bias of the turning behavior that has been reported in the literature (Hu et al., 2015; Golze et al., 2020). But by excluding turning trajectories, we significantly limit the dataset, as many intersection arms crossed exclusively by turning trajectories are completely excluded from the dataset used for constructing the classification model. As so far none of the existing research studies has investigated this issue, it seems to be an interesting aspect of the problem that needs to be clarified. The same argument applies to the minimum number of trajectories that must traverse an intersection arm, in order the latter to be included in the dataset used in the learning process. In the current experiments, the requirement was set to five trajectories (Section 4.3.6). Perhaps a larger number could further benefit the performance.

4.6. Summary

In this chapter, four TRR methods were proposed and tested on three datasets. The hybrid approach using features derived from both maps and trajectories was found to be the most effective. Its classification accuracy on the three datasets is 0.95 (Champaign), 0.82 (Chicago) and 0.88 (Hanover). Three main directions were identified for further investigation of the problem in the context of improving classification performance: 1) adapting the four methods to a feature vector that is more inclusive in terms of the information that nearby intersection arms can also contribute, 2) examine the role of turning trajectories and the number of trajectories per intersection arm on classification performance; and 3) investigate how the size of the dataset (labeled examples) affect the predictive quality of the classifiers. The next chapter deals with the first two points and Chapter 6 deals with the latter.

4.7. Acknowledgements

The methodology described in this chapter as well as its classification performance have been published in the ISPRS International Journal of Geo-Information under the title *Recognition of Intersection Traffic Regulations From Crowdsourced Data* (Zourlidou et al., 2023). The author of this thesis is the principal investigator and first author of the article, as well as the creator of all figures.

5. TRR From GPS Data: One-Arm versus All-Arm Models

5.1. Introduction

In this chapter, the four TRR methods discussed in the previous chapter are further investigated in the context of a modified feature vector that contains information from all intersection arms of the same intersection (Section 5.2.1). The latter classification models are called *all-arm* models to distinguish them from *one-arm* models explained in the previous chapter. Section 5.2.3 explores whether training separate models for three-way and four-way intersections improves the classification performance. In addition, Sections 5.2.4 and 5.2.5 discuss the role of turning trajectories and the number of trajectories per intersection arm on classification performance. A detailed analysis of the misclassification cases in the three datasets is presented in Section 5.3.6. Motivated by the results of the latter, a post-processing step is proposed to analyze and recover wrongly predicted labels as well as to predict labels from arms with missing data (Section 5.2.6). All results are presented in Section 5.3 and discussed in Section 5.4.

5.2. Methodology

5.2.1. 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 from 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 all arms of the same intersection, that from now on will be called *context arms*. For example, the *i_arm* of the intersection depicted in Figure 4.9(a) according to the all-arm c-dynamic model has 4(arms)x10(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 TRR methodologies, to the author's knowledge, none has considered such feature settings.

5.2.2. The Effect of Sampling Rate

One observation looking at the three trajectory datasets is that they have different sampling rates: Champaign 1 Hz (1 sample every 1 s), Chicago 0.28 Hz (1 sample every 3.6 s) and Hanover 0.59 Hz (1 sample every 1.7 s). The sampling rate can affect both the calculated vehicle speed and the detection of stop and deceleration events, as described in Section 4.3.1. Obviously, the higher the sampling rate, the more accurate the speed calculation is and the more movement episodes are detected. Therefore, one can assume that the sampling rate can affect the classification performance. To test this hypothesis, we conducted experiments in the two datasets with higher sampling rate, Champaign and Hanover, by undersampling the original datasets and comparing the performance in the undersampled datasets with that in the original datasets. The Champaign dataset was subsampled at ≈2 s, ≈3 s and ≈4 s. The Hanover dataset was subsampled at ≈4 s[1].

5.2.3. Reduced Models

The all-arm models, as explained in the previous section, include features from nearby intersection arms from the same intersection. The problem is that there are two types of intersections in the dataset: 3-way intersections and 4-way intersections, with three and four arms, respectively. Suppose we construct a learning model for both types of intersections (default model), in the case of arms belonging to 3-way intersections, some of the features (those corresponding to the fourth arm) will have *null* values. To clarify whether separate learning models for each type of intersection would perform better than the default model, experiments are conducted by training one classifier for arms belonging to 3-way intersections and another one for 4-way intersections.

5.2.4. The Effect of Turning/No-Turning Trajectories

Depending on the shape of the intersection they are crossing, vehicles can go straight, turn left or right. This means that the trajectory datasets contain both straight and curved trajectories. The effect of turning at an intersection generally affects the driving behavior before and after the turn compared to a straight-line driving crossing, because the vehicle must slow down before the turn and accelerate again after the turn. For this reason, other related studies have excluded curved trajectories from the dataset (Hu et al., 2015; Golze et al., 2020). By excluding these trajectories, however, the data available for building and testing a classifier is significantly reduced. In addition, at T-intersections, the arm leading to the intersections that allows only turns will always be excluded from the dataset, as its

---

[1] Since the initial sampling rate of Hanover dataset is 1.7 s, theoretically we can assume that we can undersample at 1.7x2=3.4 s. In practice though, undersampling turned out to be over 3.4 s, therefore closer to 4 s than to 3 s.

trajectory samples will always be curved. Motivated from these observations, we investigate the effect of using either all available trajectories (all combinations of right, left and straight trajectories) or exclusively straight trajectories on the classification performance (Section 5.3.4).

5.2.5. The Effect of Number of Trajectories

We consider whether there is a minimum number of trajectories per intersection arm required to apply the proposed method (Section 5.3.5). In addition, we investigate whether there is an optimal number of trajectories per intersection arm with which the classifier performs best. On the one hand, by setting a minimum number of trajectories as a condition in order for an intersection arm to be included in the classification process, we shrink the dataset: the larger this number is, the fewer intersection arms satisfy the condition, as most intersection arms have only a few trajectories. On the other hand, summarizing the behavior of the collective movement (classification features) using only a few trajectories may lead to an incorrect representation of the *real* movement behavior. This aspect of the problem is addressed by conducting experiments on the minimum number n of trajectories that an intersection arm must have to be included in the dataset used for the classification task:

(a) using all available trajectories crossing an intersection arm, from those arms that qualify the condition to be crossed by at least n trajectories; and

(b) using exactly n trajectories to compute the classification features of an intersection arm, from those arms that qualify the condition to be crossed by at least n trajectories.

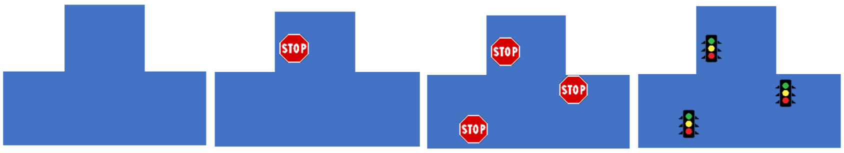
Thus, in (a), suppose we set the minimum number of trajectories to $min = 10$, we exclude from the dataset all intersection arms crossed by fewer than 10 trajectories, and compute the classification features for the remaining arms using *all* the trajectories that cross each of them. If an intersection arm now has, for example, 35 trajectories, we compute the features based on all 35 trajectories. In (b), conversely, having excluded intersection arms with fewer than 10 trajectories, we compute the features for the remaining arms using *exactly* 10 randomly sampled trajectories from each arm.

5.2.6. Application of Domain Knowledge Rules

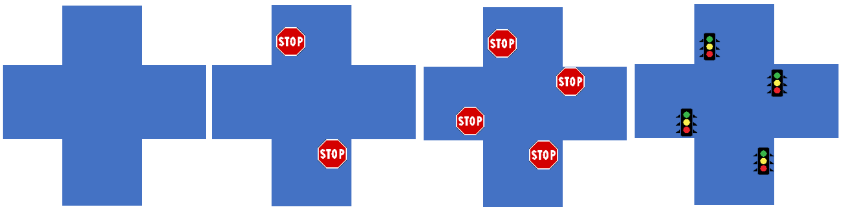
Table 5.1 shows the combinations of traffic regulators found in intersections in the three datasets. The Champaign dataset has 350 three-way intersections, 293 of which are controlled with UN-UN-SS, 33 are all-way UN, 15 are all-way SS, and 9 are all-way TS. Similar combinations are found also in four-way intersections. The Chicago dataset has similar regulation combinations as the Champaign dataset. In Hannover there are combinations of UN, PS, YS, SS and TS regulations. Similar combinations are met in four-way intersections also. One can observe from Table 5.1 that regulator categories are not randomly combined with each other, but there are underlying *rules*. To highlight such rules, we illustrate in Figure 5.1 *all* possible regulation combinations found in the datasets. We observe that a traffic signal (TS) coexists *only* with other TS in the same intersection, in all datasets. Moreover, in Champaign/Chicago, a stop sign (SS) coexists only with other SS or uncontrolled arms (UN). Furthermore, the position of the regulations in the same intersection is also not random. One can observe that *opposite* arms (arms whose angle between them is 150-200 degrees) are controlled with the same regulation. For example, in two-way stop controlled intersections, in Champaign/Chicago (Figure 5.1f), stop signs control *opposite* arms, i.e.,

Table 5.1.: Combinations of traffic regulators at intersections in the three datasets. The numbers in the cells in the first block of the table refer to number of intersections that belongs to a certain regulation combination (e.g. UN-UN-SS, PS-PS-YS, etc). In the second block the numbers refer to the percentage of intersections that belongs to a certain regulation combination out of the total number of intersections (e.g. Champ. 84% = 293/350).

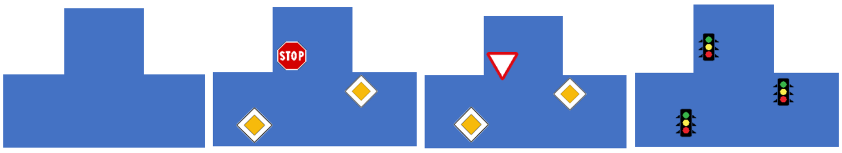
Dataset	three-way intersections							four-way intersections						
	UN UN SS	PS PS YS	PS PS SS	UN (all)	SS (all)	TS (all)	Total	UN UN SS SS	PS PS YS YS	PS PS SS SS	UN (all)	SS (all)	TS (all)	Total
Champ.	293	-	-	33	15	9	350	220	-	-	9	52	80	361
Chicago	36	-	-	4	8	8	56	10	-	-	-	17	71	98
Hanover	-	386	5	230	-	88	709	-	82	9	94	-	153	338
Champ.	84%	-	-	9%	4%	3%	100%	61%	-	-	3%	14%	22%	100%
Chicago	65%	-	-	7%	14%	14%	100%	10%	-	-	0%	17%	73%	100%
Hanover	-	55%	1%	32%	-	12%	100%	-	24%	3%	28%	-	45%	100%



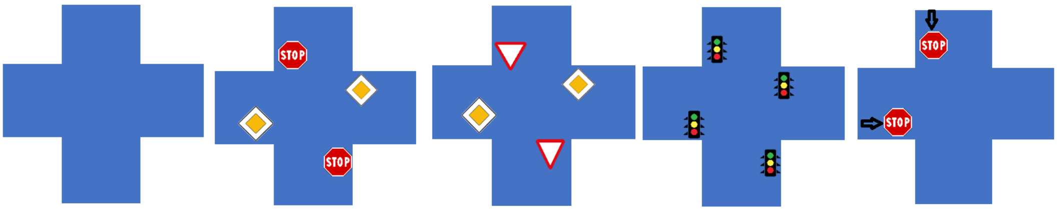
(a) All-way uncon. *(b)* One-way stop. *(c)* All-way stop. *(d)* Traffic signals.



(e) All-way uncon. *(f)* Two-way stop. *(g)* All-way stop. *(h)* Traffic signals.



(i) All-way uncon. *(j)* Priority-stop. *(k)* Priority-yield. *(l)* Traffic signals.



(m) All-way uncon. *(n)* Priority-stop. *(o)* Priority-yield. *(p)* Traffic signals. *(q)* Invalid regulation combination.

Figure 5.1.: Traffic regulation combinations in intersections: (a)-(h) valid in Champaign and Chicago, (i)-(p) valid in Hanover, (q) an example of invalid regulation combination.

there is no four-way intersection that is 2-way stop controlled and the stop signs regulate perpendicular arms such as the synthetic example shown in Figure 5.1q.

Similar rules regarding 1) the types of regulations that coexist in the same intersections and 2) their relative position with each other, are observed in Hanover dataset as well (Figures 5.1i-5.1p). Such simple *domain knowledge rules* have been used by Saremi and Abdelzaher (2015) and Hu et al. (2015), as described in page 69 and 71, but without investigating how much they contribute to overall accuracy. This thesis investigates further how such knowledge rules can contribute in increasing the classification accuracy.

One idea for making use of such knowledge rules is to examine at a post-classification step whether the predicted labels of the arms of the same intersection are in accordance with these rules, that concern both the type of regulations that are predicted for arms of the same intersection and the relative position (angle) of them with each other. More specifically, by checking whether the predicted labels agree with the knowledge rules, both wrongly predicted labels can be discovered and recovered (corrected) and at the same time predictions can be made about arms with missing data (arms for which no prediction can be made due to unavailability of trajectories).

For implementing such domain knowledge rules, this thesis takes into account the probability of predicted labels, so that only predictions with high probability (we use as threshold >0.80) are considered in the decision to recover misclassified labels. In addition, the proposed methodology goes a step further and compares the predictions of context arms, both for correcting misclassified labels and for predicting labels for arms for which there is no data (trajectories) to make predictions. In the latter case, predictions are made for arms with missing data based on the predictions of context arms for which data are available. For example, for a three-way intersection, if one arm is predicted to be TS with high probability (e.g., 0.96, which is greater that the threshold of 0.80 we have set), it can be inferred that the other two arms (for which no trajectory is available to extract features from) are also TS, making use of the knowledge rule that the regulation TS coexists only with TS in the same intersection. Another example of implementing domain knowledge checks at a post-classification step is the following. If in a three-way intersection there are two predictions for two arms, 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, too, so that the intersection complies with the domain knowledge rule that:

> *If one arm is predicted as* TS with high probability (>0.80), then all other context arms in the intersection are TS too.

We use 0.14 as the correction threshold, i.e., the difference between the two predictions. For example here the difference is 0.95-0.79=0.16, so the correction of the lower predicted label is triggered. This means that if the highest predicted label had smaller than 0.14 probability difference from the other predicted label, no correction action would be taken, on the basis that although the predictions are inconsistent with each other, their probabilities do not differ enough to decide which one to correct.

Domain knowledge rule checks for four-way intersections involve checking more conditions than in three-way intersections, as the combinations of regulations and the number of available predicted labels (one, two, three or four predicted labels) impose additional conditions to be examined. For the Champaign and Chicago dataset, an example of a domain knowledge rule check that involves examination also of the angle of the predicted regulation labels is the following:

> *If there is a total of two available predictions for a four-arm intersection and both are UN, check the angles of the predicted arms. If they are not opposite with each other and both are predicted with high probability (>0.80), then add two more UN predictions for the two missing arms (Figure 5.1e). If they are opposite, do nothing, as the two missing arms may be both either SS (Figure 5.1f) or UN (Figure 5.1e).*

Similar consistency checks shall be carried out for the other cases of predicted labels. Since Hanover contains different combinations of regulators, the domain knowledge rules differ from those of Champaign and Chicago. Due to space constraints and the intuitive nature of the domain knowledge rules, we refrain from listing the other consistency checks. The results of applying domain knowledge rules in the three datasets in a post-classification step are given in Section 5.3.7.

5.2.7. Classification Settings

Two tree-based classifiers are used to classify the intersection arms: the RF classifier and the GB classifier. The XGBoost (XGBoost Python, 2022) library was used for the implementation. All programming work has been implemented in Python 3.7. Unless otherwise stated, default feature settings are assumed to be features extracted from straight trajectories, from intersections containing at least five trajectories. Intersection arms that are crossed with fewer than five trajectories are excluded from training and testing in the default model.

5.3. Results

This section presents all the classification results of the experiments discussed in the previous Section (5.2). First the performance of all-arm classification models is presented (Section 5.3.1). The best model is then used for all other experiments: testing the performance of reduced models in Section 5.3.3, testing the effect of turning trajectories on classification performance in Section 5.3.4, testing the effect of number of trajectories in Section 5.3.5, conducting a misclassification analysis in Section 5.3.6 and evaluating the application of domain knowledge rules in Section 5.3.7.

5.3.1. One-arm vs. All-arm Models

Table 5.2 shows the classification accuracy and F1-score of one-arm and all-arm methods for the two classifiers (RF and GB). We can see that GB classifier performs as well or better than RF for almost all experiments. Only in the one-arm c-Dynamic model for the Chicago and Hannover dataset, RF performs slightly better than GB (+0.1 accuracy).

With respect to all-arm models, we observe that the static model performs much better than the respective 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 than the all-arm static model. 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. Similar results are observed 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

Table 5.2.: Classification accuracy (Acc) and F1-score (F_1) of the TRR models. Highlighted are the best one-arm and all-arm models for each dataset.

	Method	Champaign				Chicago				Hanover			
		RF		GB		RF		GB		RF		GB	
		Acc	F_1	Acc	F_1	Acc	F_1	Acc	F_1	Acc	F_1	Acc	F_1
One-arm	Static	0.67	0.67	0.69	0.69	0.72	0.71	0.72	0.71	0.61	0.61	0.62	0.62
	c-Dynamic	0.93	0.93	0.93	0.93	0.78	0.78	0.77	0.77	0.86	0.86	0.85	0.84
	Dynamic	0.94	0.94	0.94	0.94	0.81	0.81	0.81	0.81	0.86	0.87	0.87	0.87
	Hybrid	0.94	0.94	0.95	0.95	0.82	0.82	0.82	0.82	0.87	0.87	0.88	0.89
All-arm	Static	0.86	0.86	0.86	0.86	0.89	0.89	0.89	0.89	0.86	0.86	0.87	0.87
	c-Dynamic	0.94	0.94	0.94	0.94	0.78	0.78	0.78	0.78	0.89	0.89	0.90	0.90
	Dynamic	0.94	0.94	0.95	0.95	0.83	0.83	0.84	0.84	0.90	0.90	0.91	0.91
	Hybr.-all static*	0.94	0.94	0.95	0.95	0.88	0.87	0.91	0.91	0.91	0.91	0.95	0.95
	Hybr.-all dynamic$^\diamond$	0.95	0.95	0.95	0.95	0.82	0.82	0.86	0.85	0.91	0.91	0.93	0.93
	Hybrid$^\nabla$	0.95	0.95	0.95	0.95	0.88	0.87	0.90	0.90	0.92	0.91	0.95	0.95

* Only the dynamic features from one arm are included, along with the static features from all intersection arms of the intersection.

$^\diamond$ Only the static features from one arm are included, together with the dynamic features from all intersection arms of the intersection.

$^\nabla$ The method uses the dynamic features from the adjacent intersection arms and the static features of all the intersection arms of the intersection.

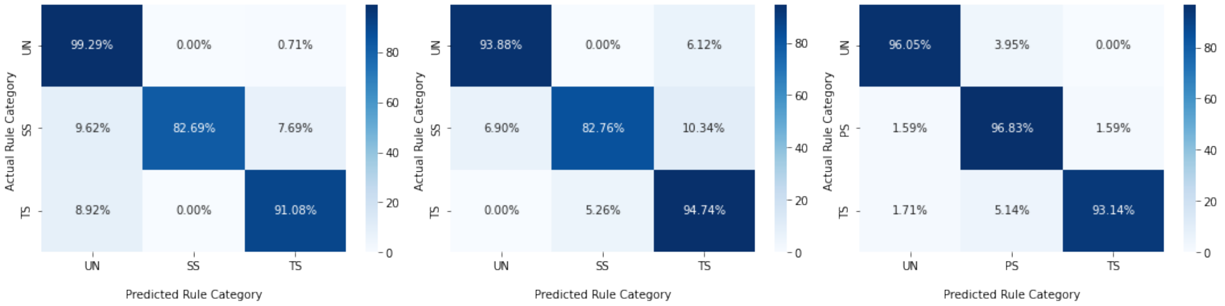
the GB classifier performs better for all datasets and for this reason this model is selected to be used as the *default* model for the experiments in the following sections.

In addition, feature selection and parameter tuning was performed for the all-arm hybrid-all-static model. In Appendix A.1, plots are provided 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). 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 along with map features. This fact we can say that was expected, as in Champaign the dynamic method already performed very well (without using any static features, compare (cf.) dynamic and hybrid classification accuracy in one-arm and all-arm models in Table 5.2). Common significant features for all datasets are the pattern features ($p1$, $p2$, $p3$ and $p4$). In the Hanover dataset there are more important features compared to the other two datasets, from which many are map features (obtained from OSM). Only in the Champaign dataset, the map-based features are less important compared to the Hanover and Chicago datasets. The classification results and confusion matrices for the three datasets after feature selection and parameter tuning are presented in Table 5.3 and Figure 5.2. A detailed classification report with per class performances can be found in Appendix A.2.

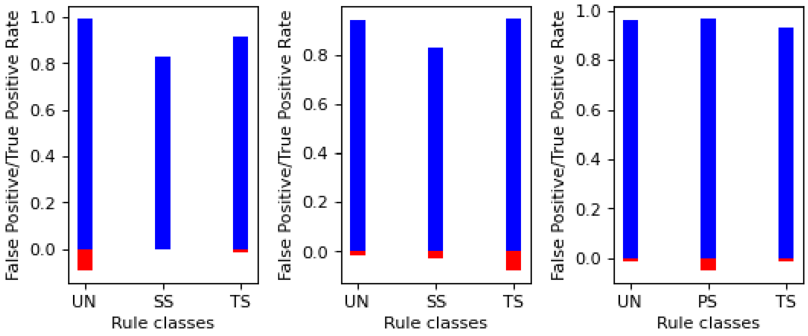
As we can see in Table 5.3, feature selection and parameter tuning increased the accuracy by 1%, from 0.95 to 0.96 for the Champaign and Hanover datasets, and from 0.91 to 0.92 for the Chicago dataset. In Champaign and Chicago (see Appendix A.2), the stop sign (SS) category is predicted slightly worse 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

Table 5.3.: Classification performance of the hybrid all-static model (GB).

	Dataset	Recall	Precision	F-score	Accuracy
Default param.	Champaign	0.95	0.95	0.95	0.95
	Chicago	0.91	0.92	0.91	0.91
	Hanover	0.95	0.95	0.95	0.95
After tuning	Champaign	0.96	0.96	0.96	0.96
	Chicago	0.92	0.93	0.92	0.92
	Hanover	0.96	0.96	0.96	0.96



(a) Champaign. *(b)* Chicago. *(c)* Hanover.



(d) Champaign. *(e)* Chicago. *(f)* Hanover.

Figure 5.2.: Confusion matrices and false/true positive rates for the three datasets.

also highlighted in the confusion matrices in Figure 5.2, which visually depicts the actual versus predicted classes. In the same figure, there are also graphs of the FPR and TPR. We can see in Figures 5.2(d), (e) and (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). Also, the highest TPRs are observed in the same classes as the highest FPRs: Champaign: 0.99 (UN), Chicago: 0.95 (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) could be possibly 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 4.2.2, the sampling rate in Chicago is lower than the other two datasets, which may affect the computation of the feature calculation (short-term detected episodes).

5.3.2. Testing the Effect of Sampling Rate

Table 5.4 shows the classification performance of TRR methods at different sampling rates. In 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 1-3%. Regarding the detected stop and deceleration episodes, it seems that the sampling rate affects them: 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 2-3%, 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 2-12%, with the smallest difference seen in the hybrid all-static model.

A general conclusion from these experiments is that sampling rate can affect classification performance: in the Champaign dataset, when 1s 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 may affect the training of the classifier. Also the fact that the GPS tracks in Chicago are collected from shuttle buses can be another possible factor that affects the classification, if for example, we assume that there are bus stops near intersections, and stop episodes are initiated due to bus stops additionally to those initiated from intersection traffic regulations.

Table 5.4.: Classification accuracy of the TRR methods under different sampling rates (undersampling). The original datasets are highlighted in grey.

		Champaign				Chicago	Hanover	
		$\approx$ 1 s	$\approx$ 2 s	$\approx$ 3 s	$\approx$ 4 s	$\approx$ 4 s	$\approx$ 2 s	$\approx$ 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	5,589	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
	Hybrid (all-static)	0.95	0.95	0.94	0.93	0.91	0.95	0.94

5.3.3. Reduced Models

Table 5.5 shows the classification performance of the reduced models, as explained in Section 5.2.3, as well as that of the default model. The detailed classification report with performance per class is provided in Appendix A.3. On all datasets the default model performs as well or better than the reduced models in terms of F-score. Therefore, in all upcoming experiments in this thesis the default model will be used.

Table 5.5.: Classification performance of the reduced and default models.

Dataset	Recall	Precision	F-score	Accuracy
Champaign (3-way)	0.96	0.96	0.96	0.96
Champaign (4-way)	0.96	0.96	0.96	0.96
Champaign (default)	0.96	0.96	0.96	0.96
Chicago (3-way)	0.92	0.88	0.89	0.92
Chicago (4-way)	0.93	0.94	0.92	0.93
Chicago (default)	0.92	0.93	0.92	0.92
Hanover (3-way)	0.95	0.96	0.95	0.95
Hanover (4-way)	0.96	0.97	0.96	0.96
Hanover (default)	0.96	0.96	0.96	0.96

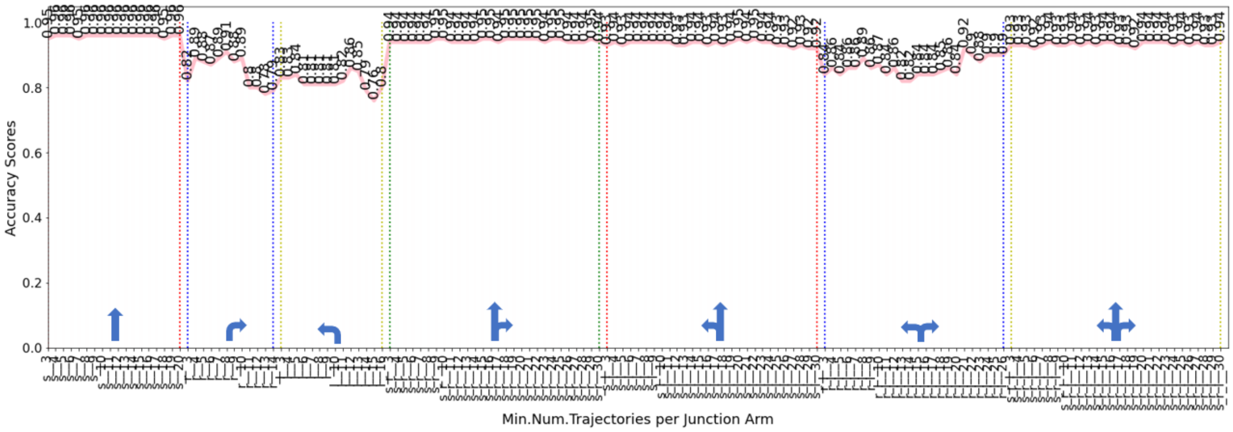
5.3.4. Testing the Effect of Turning Trajectories and Examining an Optimal Number of Trajectories

Figure 5.3 depicts graphs with the classification performance (accuracy) 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.

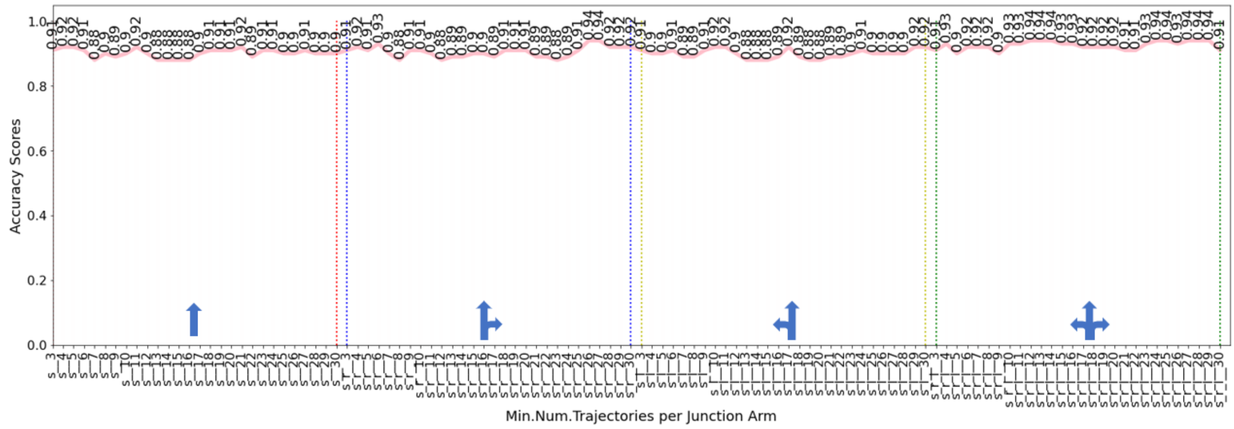
We looked at all possible turning settings and their combinations: straight trajectories ($s\_$ in Figure 5.3), trajectories that turn right ($r\_$), left ($l\_$), straight and right turn ($s\_r\_$), straight and left ($s\_l\_$), right and left ($r\_l\_$) and straight, right and left turning trajectories ($s\_r\_l\_$). The number after these prefixes in the figure refers to the number n of trajectories used to select the intersection arms (minimum number of trajectories per intersection arm) and to calculate the classification features. Not all turning settings are tested with the same number of trajectories, because for each turning/crossing setting, we require the test data set to contain at least seven intersection arms per class. E.g., in the Champaign dataset (Figure 5.3a), we tested the straight trajectories for various numbers 3, 4,..., 20, because for minimum number of trajectories equal to 21, the number of intersection arms in the test set (10-fold cross-validation) did not contain more than seven stop controlled (SS) intersection arms. SS is the type of regulation with the fewer examples in this dataset, and we set such a number to ensure that the classifier will be tested to at least seven SS examples.

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 5.3a and 5.3b). 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, 2,619 right and 2,768 left, in Hanover 19,092 straight, 3,394 right and 3,073 left, in Chicago 12,638 straight, 2,820 right and 1,301 left).

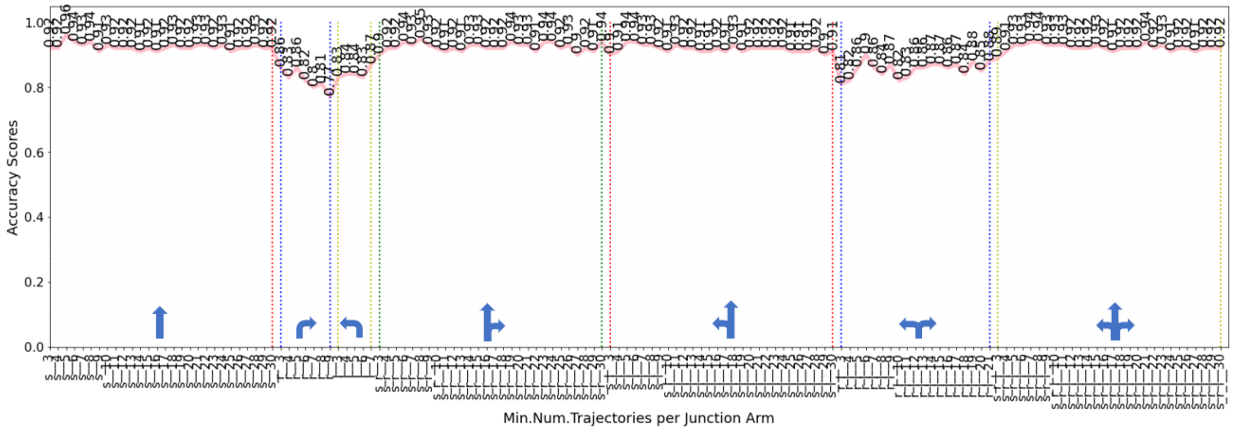
A possible explanation for the poor classification performance when using exclusively turning trajectories (e.g., in Champaign 0.88 accuracy for right turning trajectories when intersection arms have at least 5 trajectories - see r_5 at Figure 5.3a), is the smaller dataset used for training compared to the other settings. Table 5.6 shows the number of intersection arms per



(a) Champaign.



(b) Chicago.



(c) Hanover.

Figure 5.3.: 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).

control type in Champaign dataset, when each arm has at least five trajectories. We see that the datasets when only right, left or right/left trajectories are used are much smaller than the dataset with straight trajectories. Additionally the fact that features are computed from a few trajectories (as the number of right and left trajectories per intersection 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 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 intersection arms sampled from a higher density of turning trajectories would not show such a strong negative effect of turning trajectories.

Table 5.6.: Dataset size for different trajectory direction settings with minimum number of trajectories per intersection arm equal to 5 (Champaign dataset).

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

Another observation is that in the Champaign dataset the performance when only straight trajectories are used is better than when straight and turning trajectories are used. E.g., with at least 5 straight trajectories the accuracy is 96% and with at least 5 straight/turning trajectories the accuracy is 93% (Figure 5.3a). In the Hanover dataset (Figure 5.3c), 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%-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 5.3b). Although there are not a sufficient number of intersection 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. E.g. 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 trajectories 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 intersection arm equal to 5), excluding all curved tracks at intersections. The arms of T-intersections that are crossed from trajectories that always need to turn at the intersection are affected from this condition, as they will be always excluded from the datasets used for classification. Nevertheless, as discussed in Section 5.2.6, the regulators of these arms can be recovered with the help of the predicted labels of their context arms and the usage of domain knowledge rules.

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

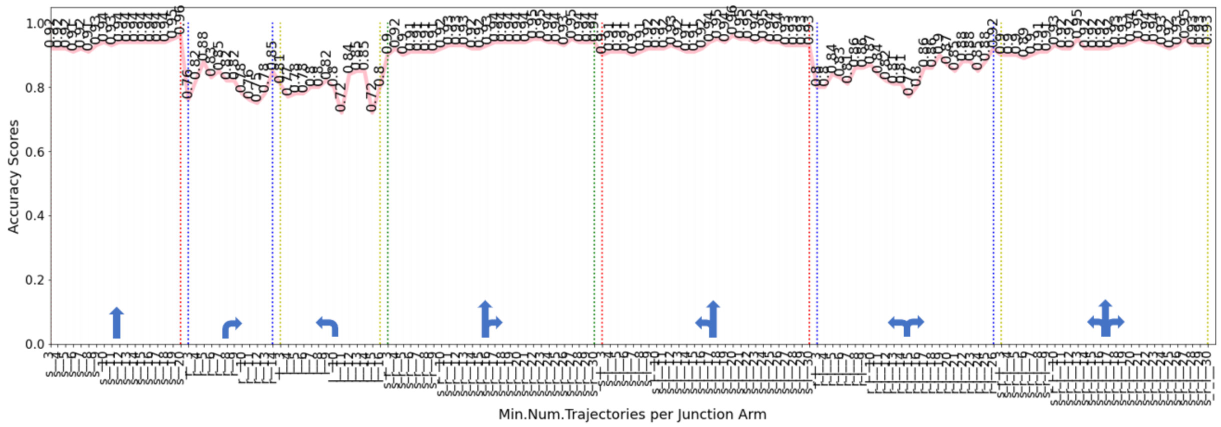
The number of trajectories used as a minimum requirement for the calculation of classification features, as well as for the selection of intersection arms for training and testing, seems to affect performance (Figure 5.4). Comparing the case of using a *certain* number of trajectories, that is (i.e.), a subset of all trajectories crossing an intersection arm, with the case of using *all* available trajectories per intersection arm (as explained in Section 5.2.5), the latter case achieves better performance on average. This result holds for all datasets (cf. Figure 5.3 with Figure 5.4) 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 4.3.4), the better the latter reflect the actual movement behavior.

In the Champaign dataset we see from Figure 5.4a that with just 3 straight trajectories per intersection 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 5.3a, we see that limiting the number of trajectories per arm yields lower classification performance. E.g. with *at least* 4 straight trajectories per intersection arm the accuracy is 96%.

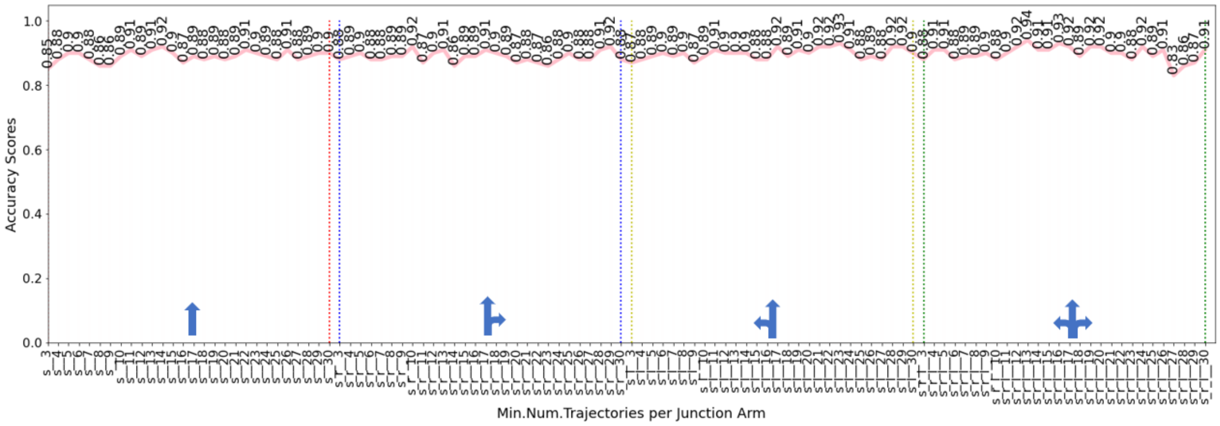
In the Hanover dataset, we can also see from Figure 5.3c that when all available traces are used the performance is better than using a certain number (Figure 5.3c). For accuracy above 91% at least 7 traces per intersection arm are required (Figure 5.4c). The best accuracy 93% is achieved with 9 trajectories. 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 5.3c).

In 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 5.4b vs. Figure 5.3b). 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 5.4b).

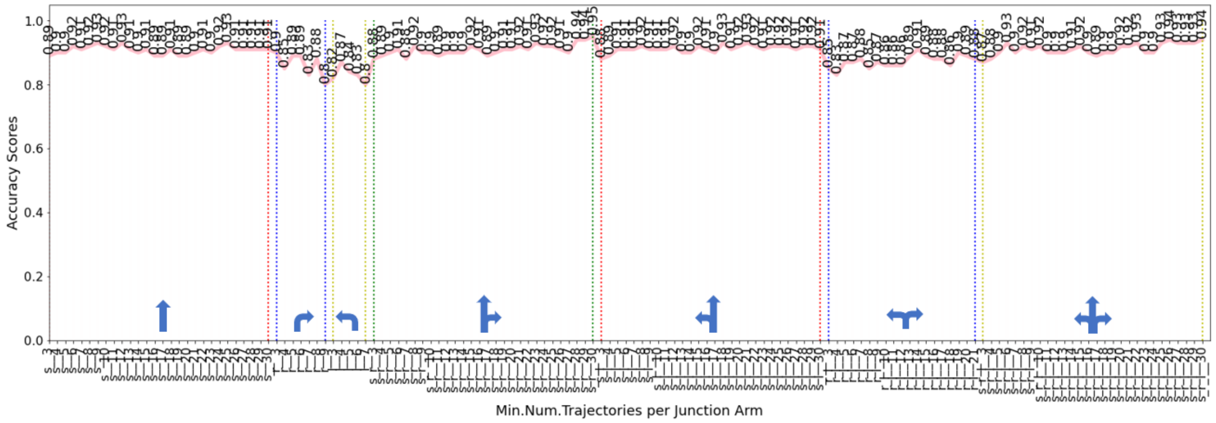
Therefore, for feature computation, excluding the number of trajectories to a certain number negatively affects the classification performance. However, with only 3 straight trajectories per intersection arm, the classification accuracy is equal to or greater than 85% across all datasets (85% in Chicago,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

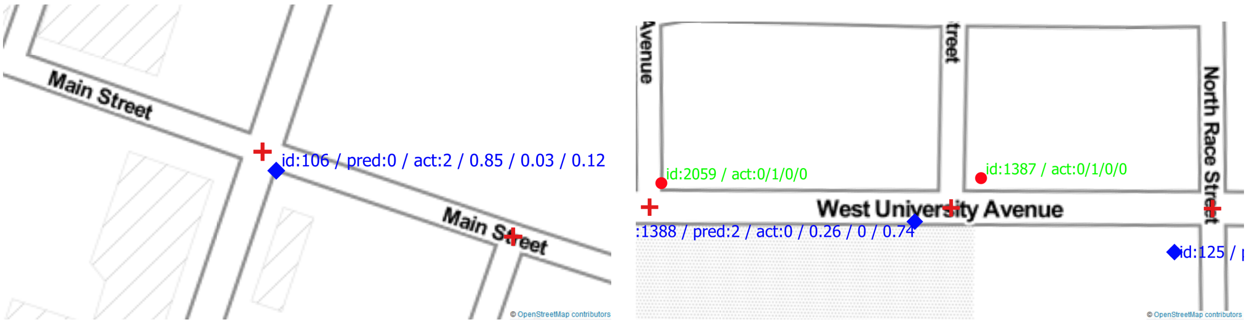
Figure 5.4.: Experiments with different number of trajectories where classification features are computed using a certain number of trajectories, i.e., 3, 4, ..., and not all available crossing trajectories.

5.3.6. Misclassification Analysis

This section analyses the incorrect predictions in the three datasets. For each dataset, all misclassification cases are documented in a table, which provides information on the probability of the (incorrectly) predicted labels, the possibility of recovering the incorrect labels using information from context arms, and the rationale for the latter.

5.3.6.1. Misclassification cases in Champaign Dataset

From the 26 wrong predictions (misclassifications) shown in Table 5.7, only 4 predicted regulation labels can be recovered (arms/regulations with identifiers 331, 1388, 1586 and 2019, shown in blue in Table 5.7) using information from context arms. These 4 regulations belong to intersections where at least one more prediction is available from the context arms, they are predicted with much lower probability than the context arm(s), and their predicted labels contradict a domain knowledge rule (e.g., a TS cannot coexist with a SS in the same intersection). For example, regulation 331 is predicted as TS with probability 0.4, while its perpendicular context arm is predicted as SS with probability 0.91, and their predicted labels contradict the domain knowledge rules (TS and SS cannot coexist in the same intersection). Therefore, the label of arm 331 is retrieved (corrected to SS) based on the predicted label of its context arm (SS), which is predicted with high probability (0.91, which is greater than the threshold of 0.80, explained in Section 5.2.6). Another "retrievable" or "recoverable" example is shown in Figure 5.5b.


Main Street
id:106 / pred:0 / act:2 / 0.85 / 0.03 / 0.12
Main Street
© OpenStreetMap contributors
Avenue
treet
North Race Street
id:2059 / act:0/1/0/0
id:1387 / act:0/1/0/0
West University Avenue
:1388 / pred:2 / act:0 / 0.26 / 0 / 0.74
id:125 /
© OpenStreetMap contributors

(a) Intersection arm id: 106 (non recoverable). *(b)* Intersection arm id: 1388 (recoverable).

Figure 5.5.: Two examples of misclassified intersection regulations from the Champaign dataset. With blue diamond is depicted an incorrectly predicted regulation. A red point depicts a correctly predicted regulation. The red cross depicts the intersection center. The label of the predicted regulations contains the following information: id number of the intersection arm/ predicted label (only for the wrongly predicted regulations)/actual label/ predicted probability of label 0/ predicted probability of label 1/ predicted probability of label 2.

As for the predicted regulation labels that cannot be recovered, in general, arms without context information (Num. adj. equals to 0, in Table 5.7) cannot be subjected to any subsequent processing (such an example is shown in Figure 5.5a). There are 4 such arms (106, 1692, 2176 and 2358, shown in red in Table 5.7). In addition, 9 arms are misclassified with high probability (arms 125, 394, 553, 760, 760, 1237, 1353, 1368, 1500 and 1880 shown in green in Table 5.7), having context arms (1 context arm each) correctly classified with also high probability (exception is the arm 760[2]), therefore the information from the

---

[2] In this case the context arm of the wrongly predicted arm with id 760, which is correctly predicted with low probability (0.61) will be (incorrectly) corrected. This is an inevitable flaw of the recovering process.

Table 5.7.: The wrong predictions in the Champaign dataset (Rec.: whether the regulation is recoverable or not, Num.Adj.: number of arms from the same intersection having predicted labels (context arms).

	Arm id.	Int. Type	Rec.	Prob.	Num. Adj.	Reasoning for (not) recovering
1	106	4-way	No	0.85	0	no predictions available from context arms
2	125	4-way	No	0.97	1	context rule is predicted with prob. 0.92
3	331	4-way	Yes	0.40	1	context rule is predicted with prob. 0.91
4	394	3-way	No	0.99	1	context rule is predicted with prob. 1.0
5	553	3-way	No	0.99	1	context rule is predicted with prob. 0.97
6	760	3-way	No	1.0	1	context rule is predicted with low prob. 0.61[1]
7	1237	3-way	No	0.99	1	context rule is predicted with prob. 1.0
8	1353	4-way	No	1.0	2	context rules are predicted with prob. 0.94 and 0.87
9	1368	4-way	No	0.94	1	context rule is predicted with prob. 0.99
10	1388	3-way	Yes	0.74	1	context rule is predicted with prob. 1.0
11	1500	4-way	No	1.0	1	context rule is predicted with prob. 0.94
12	1530	3-way	No	1.0	1	context rule (1531) is predicted (wrongly) with prob. 0.85
13	1531	3-way	No	0.85	1	context rule (1530) is predicted (wrongly) with prob. 1.0
14	1562	4-way	No	0.81	1	context rule (1564) is predicted (wrongly) with prob. 0.93
15	1564	4-way	No	0.93	1	context rule (1562) is predicted (wrongly) with prob. 0.81
16	1586	4-way	Yes	0.85	1	context rule is predicted with prob. 0.99
17	1692	4-way	No	1.0	0	no predictions available from context arms
18	1880	3-way	No	1.0	1	context rule is predicted with prob. 0.98
19	1985	4-way	No	0.99	1	context rule (1987) is predicted (wrongly) with prob. 0.96
20	1987	4-way	No	0.96	1	context rule (1985) is predicted (wrongly) with prob. 0.99
21	2019	3-way	Yes	0.48	2	context arms are predicted with the same label (UN)
22	2174	3-way	No	0.66	1	context rule is predicted with low prob. (0.79) too
23	2176	4-way	No	0.87	0	no predictions available from context arms
24	2288	4-way	No	0.98	1	context rule (2290) is predicted (wrongly) with prob. 0.53
25	2290	4-way	No	0.53	1	context rule (2288) is predicted (wrongly) with prob. 0.98
26	2358	4-way	No	1.0	0	no predictions available from context arms

context arms cannot further clarify the case. It would help, however, if there were available more than one predictions from context arms, so that the information from two or three context arms could be compared to the predicted label under examination and validate whether the predicted regulation label contradicts the predicted regulation labels of the context arms. In addition, 8 arms, from 4 intersections (2 arms from the same intersection, therefore 2x4intersections=8) could not be retrieved (1530, 1531, 1562, 1564, 1985, 1987, 2288 and 2290, non highlighted in the table) because all context arms were misclassified and their predicted labels did not contradict the domain knowledge rules, therefore no retrieval action could be triggered. Finally, one arm (2174, shown in yellow in Table 5.7) could not be retrieved, as the context arm, although correctly predicted, was predicted with low probability (0.79).

In total, from the 26 incorrect predictions, 4 predicted regulation labels (blue) are recoverable and 22 unrecoverable. From the latter, 4 arms (red) cannot be corrected using domain knowledge rules because no information from context arms is available. 9 arms (green) are incorrectly predicted with high probability, and having only one context arm (or two for 4-way intersections) correctly predicted cannot help recover the correct label (if more predictions from context arms were available, would possibly make these labels recoverable). 8 arms cannot be recovered (white), as they belong to intersections where all arms are incorrectly predicted and their predicted labels do not contradict any domain knowledge rule. Finally, 1 arm (yellow) is incorrectly predicted with low probability, but the context arm is correctly predicted with also low probability, therefore no recovery action can triggered.

5.3.6.2. Misclassification cases in Chicago Dataset

From the 12 incorrectly predicted regulation labels, 2 regulations (arms with id 83 and 419, shown in blue in Table 5.8) can be recovered with the help of domain knowledge rules. Such an example is illustrated in Figure 5.6 where the regulation with id 419 (blue diamond) is predicted as TS (pred: 2) with (low) probability 0.75, whereas its actual label is SS (act: 1). Because the predicted labels of its context arms (red points) are predicted with high probability as SS (act: 1, regulation with id 416 is predicted with probability 1 and regulation with id 417 with 0.89) and because these two regulations are perpendicular to each other, it can be inferred that the intersection is all-way stop controlled and therefore arm 419 is corrected to SS.

Table 5.8.: The wrong predictions in the Chicago dataset. (Rec.: whether the regulation is recoverable or not, Num.Adj.: number of arms from the same intersection having predicted labels (context arms).

	Arm id.	Int. Type	Rec.	Prob.	Num. Adj.	Reasoning for (not) recovering
1	83	4-way	Yes	0.63	1	context rule is predicted with prob. 0.91
2	207	4-way	No	0.96	0	no predictions available from context arms
3	215	3-way	No	0.96	0	no predictions available from context arms
4	318	4-way	No	0.90	0	no predictions available from context arms
5	337	3-way	No	0.92	1	context rule (338) is predicted (wrongly) with prob. 0.89
6	338	3-way	No	0.89	1	context rule (337) is predicted (wrongly) with prob. 0.92
7	379	3-way	No	0.99	0	no predictions available from context arms
8	385	4-way	No	0.73	0	no predictions available from context arms
9	419	4-way	Yes	0.75	2	context rules are predicted with prob. 1.0 and 0.89
10	427	3-way	No	0.63	0	no predictions available from context arms
11	478	4-way	No	0.97	1	context rule is predicted with prob. 0.94
12	554	3-way	No	0.96	0	no predictions available from context arms

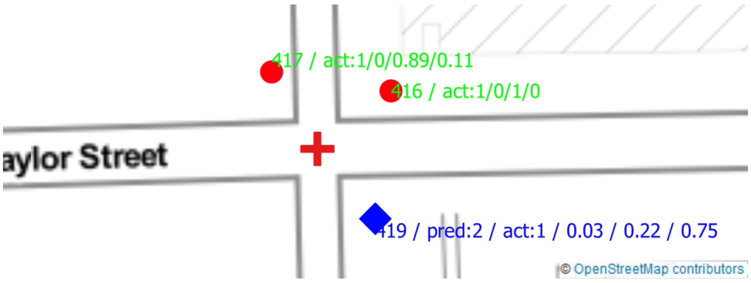

417 / act:1/0/0.89/0.11
416 / act:1/0/1/0
aylor Street
419 / pred:2 / act:1 / 0.03 / 0.22 / 0.75
© OpenStreetMap contributors

Figure 5.6.: An example of a misclassified regulation, depicted as blue diamond, which is predicted as TS (pred:2) with probability 0.75, while its actual label is SS (act:1). This regulation is recoverable due to the information from the predicted labels of its context arms, which are predicted as SS, with probabilities 1.0 and 0.89. Both probabilities are considered high, according to the defined threshold, and because the arms are perpendicular to each other, it can be inferred that the intersection is all-way stop-controlled. The regulation with id 419 is then corrected to SS.

7 regulations cannot be retrieved due to lack of available information from the context arms (arms with identifiers 207, 215, 215, 318, 379, 385, 427 and 554, shown in red in Table 5.8). In addition, 2 arms belonging to the same intersection (337 and 338, not highlighted in Table 5.8) are incorrectly predicted with high probability and their predicted labels do not contradict any domain knowledge rule, therefore no recovery action can be triggered. Finally, 1 arm (478, shown in green in Table 5.8) cannot be recovered because both the arm and its context arm are predicted with high probability.

5.3.6.3. Misclassification cases in Hanover Dataset

From the 25 incorrectly predicted regulation labels, 9 regulations (arms with ids 63, 258, 1206, 1266, 2695, 2897, 3065, 3356 and 3519, shown in blue in Table 5.9) can be recovered. 5 rules cannot be retrieved due to lack of information available from context arms (132, 327, 1093, 1317 and 1840, shown in red in the table). In addition, 2 arms belonging to the same intersection (438 and 440, not highlighted in the table) are misclassified with high probability and their predicted labels do not contradict any domain knowledge rule, therefore no recovery action can be triggered.

Table 5.9.: The wrong predictions in the Hanover dataset. (Rec.: whether the regulation is recoverable or not, Num.Adj.: number of arms from the same intersection having predicted labels (context arms).

a/a	Arm id.	Int. Type	Rec.	Prob.	Num. Adj.	Reasoning for (not) recovering
1	63	3-way	Yes	0.59	1	context rule is predicted with prob. 1.0
2	132	3-way	No	0.97	0	no predictions available from context arms
3	237	4-way	No	0.90	1	context rule is predicted with prob. 0.99
4	258	4-way	Yes	0.51	1	context rule is predicted with prob. 0.90
5	327	4-way	No	0.95	0	no predictions available from context arms
6	438	4-way	No	0.99	1	cont. rule (440) is predicted UN (wrongly) with prob. 0.51
7	440	4-way	No	0.51	1	cont. rule (438) is predicted UN (wrongly) with prob. 0.99
8	577	4-way	No	1.0	1	context rule is predicted with prob. 0.89
9	987	4-way	No	0.98	1	context rule is predicted with prob. 1.0
10	1030	3-way	No	0.97	1	context rule is predicted with prob. 0.97
11	1093	3-way	No	0.94	0	no predictions available from context arms
12	1136	4-way	No	0.83	1	context rule is predicted with prob. 0.96
13	1206	4-way	Yes	0.63	1	context rule is predicted with prob. 0.99
14	1266	4-way	Yes	0.70	1	context rule is predicted with prob. 0.96
15	1317	3-way	No	0.84	0	no predictions available from context arms
16	1323	4-way	No	0.96	1	context rule is predicted with prob. 0.90
17	1840	3-way	No	0.85	0	no predictions available from context arms
18	2695	4-way	Yes	0.65	1	context rule is predicted with prob. 1.0
19	2872	3-way	No	1.0	1	context rule is predicted with prob. 0.92
20	2875	3-way	No	1.0	1	context rule is predicted with prob. 1.0.
21	2897	3-way	Yes	0.74	1	context rule is predicted with prob. 0.99.
22	3065	3-way	Yes	0.52	1	context rule is predicted with prob. 1.0.
23	3134	3-way	No	0.98	1	context rule is predicted with prob. 1.0.
24	3356	3-way	Yes	0.56	1	context rule is predicted with prob. 1.0.
25	3519	4-way	Yes	0.52	1	context rule is predicted with prob. 0.85.

Finally, 8 arms (577, 987, 1030, 1136, 1323, 2872, 2875 and 3134, marked in green) cannot be recovered because both the arm and its context arm are predicted with high probability. Such an example is depicted in Figure 5.7, where both (opposite) arms are predicted with high probabilities with labels that contradict domain knowledge rules, i.e. the arm with id 1323 (blue) is predicted as PS (pred:1) with probability 0.96 and the other (opposite) arm with id 1322 as UN (act:0) with probability 0.90. Because both are predicted with high probability (larger than the defined threshold of 0.80), no decision can be taken on which regulation to correct.

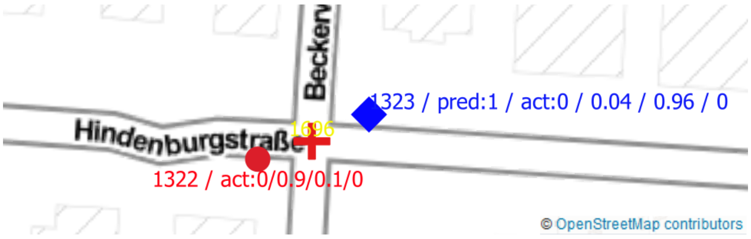

Beckerw
Hindenburgstraße
1323 / pred:1 / act:0 / 0.04 / 0.96 / 0
1696
1322 / act:0/0.9/0.1/0
© OpenStreetMap contributors

Figure 5.7.: An example of an incorrectly predicted regulation (id:1323) from the Hanover dataset which is non recoverable, due to the fact that both context arms are predicted with high probabilities (arm 1323 with 0.96 and arm 1322 with 0.90). The label of the predicted regulations contains the following information: id number of the intersection arm/ predicted label (only for the wrongly predicted regulations)/actual label/ predicted probability of label 0/ predicted probability of label 1/ predicted probability of label 2.

5.3.7. Applying Domain Knowledge Rules

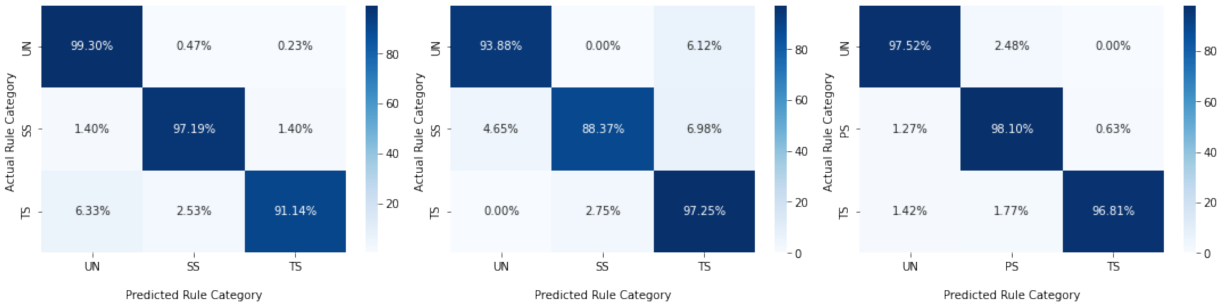
Table 5.10[3] shows the classification report after applying the domain knowledge rule consistency check, as explained in Section 5.2.6. Figure 5.8 also shows the confusion matrices along with the FPR/TPR graphs for the three datasets. The first observation from Table 5.10 is that the accuracy increases by 1% in Champaign and Hanover (from 96% to 97%) and by 3% in Chicago (from 92% to 95%). 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, PS and TS are 1.3%, 2%, 0.5%. Compared to the FPRs before applying the knowledge rules (cf. Figure 5.2 with 5.8), 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%)

---

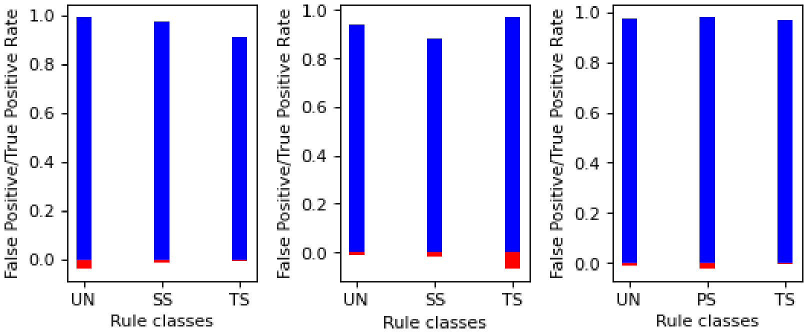
[3] A detailed classification report with performance by class is given in Appendix A.4.

Table 5.10.: Classification results of the default model before and after applying domain knowledge rules for recovering incorrect predictions and predicting regulation labels, when possible, for arms with no available trajectory data.

	Dataset	Recall	Precision	F-score	Accuracy
Default	Champaign	0.96	0.96	0.96	0.96
	Chicago	0.92	0.93	0.92	0.92
	Hanover	0.96	0.96	0.96	0.96
After applying knowledge rules	Champaign	0.97	0.97	0.97	0.97
	Chicago	0.94	0.94	0.94	0.95
	Hanover	0.98	0.97	0.98	0.97



(a) Champaign. *(b)* Chicago. *(c)* Hanover.



(d) Champaign. *(e)* Chicago. *(f)* Hanover.

Figure 5.8.: Confusion matrices and false/true positive rates for the three datasets after applying consistency checks using domain knowledge rules.

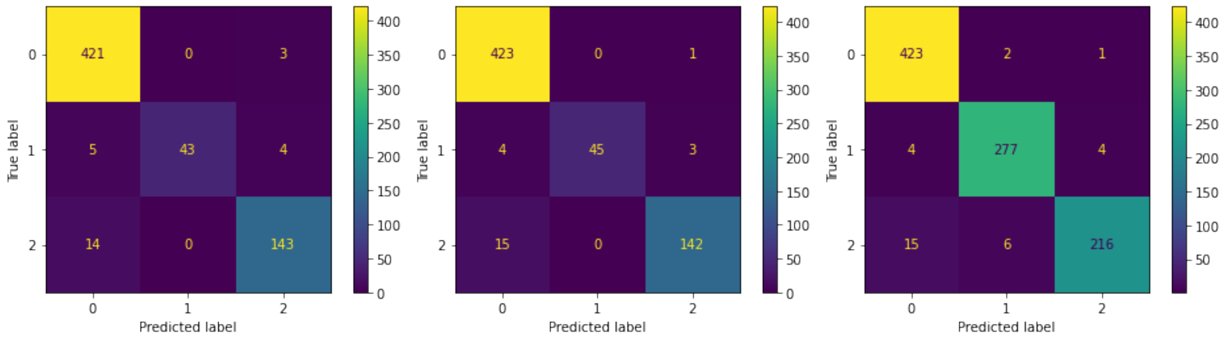
and for SS decreased from 3.2% to 1.8% (44% decrease), and in Hanover the FPR for PS decreased from 4.8% to 2% (58.3% decrease). Moreover, the average FPR (TPR) across all regulator classes decreased (increased) in Champaign from 3.5% (91%) to 1.8% (95.8%), in Chicago from 4.3% (90.5%) to 3.2% (93.2%) and in Hanover from 2.6% (95.3%) to 1.3% (97.5%) respectively.

Figure 5.9 shows the confusion matrices before the domain knowledge rules are applied (Figures 5.9a, 5.9d, 5.9g), after applying the domain knowledge rules to retrieve *only* possible incorrectly predicted regulations (Figures 5.9b, 5.9e, 5.9h) and after both recovering possible wrongly predicted labels and predicting labels from missing arms (Figures 5.9c, 5.9f and 5.9i). Both recovering incorrect predictions and predicting labels for arms with missing data are done in the same step, i.e., in the same consistency check-step an arm is checked if its predicted label is consistent with the domain knowledge rules, a decision is made, when possible, for predicting regulation labels for the context arms with missing data.

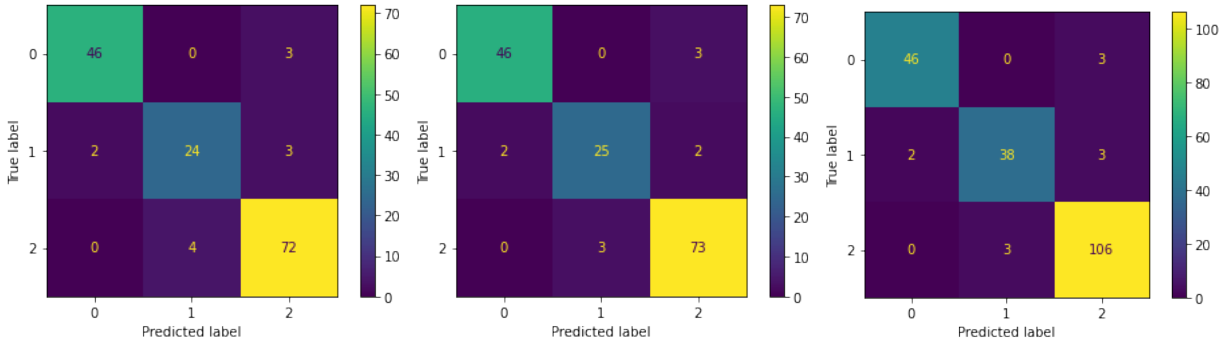
In Champaign we can see from Figures 5.9a and 5.9b, that the 4 recoverable regulators shown in Table 5.7 were recovered (UN from 421 to 423 and SS from 43 to 45) and that 1 regulator was recovered incorrectly (a TS was changed to UN, cf. last row in Figure 5.9b from 143, which was the correct predicted number of TS in Figure 5.9a, was decreased to 142), as an inevitable result of the application of domain knowledge rules that discussed in page 112. In total, 310 predictions were made from arms with missing data (cf. Figure 5.9b and 5.9c), of which only 4 were incorrect (99% accuracy). This means that the predictions for the context arms with no data available, are not only accurate, but lead to an increase in predicted arms equal to 49% of the original dataset without using any data (original dataset: 638 arms, after applying domain knowledge rules: 948 predictions).

In Chicago, the 2 recoverable regulators shown in Table 5.8 were recovered (cf. Figure 5.9d and 5.9e, the true positives increased from 142 to 144). In addition, 47 new predictions were made on arms with missing data, representing a 30.5% increase of the original dataset (cf. Figure 5.9e and 5.9f), of which only one was incorrect (98% accuracy).

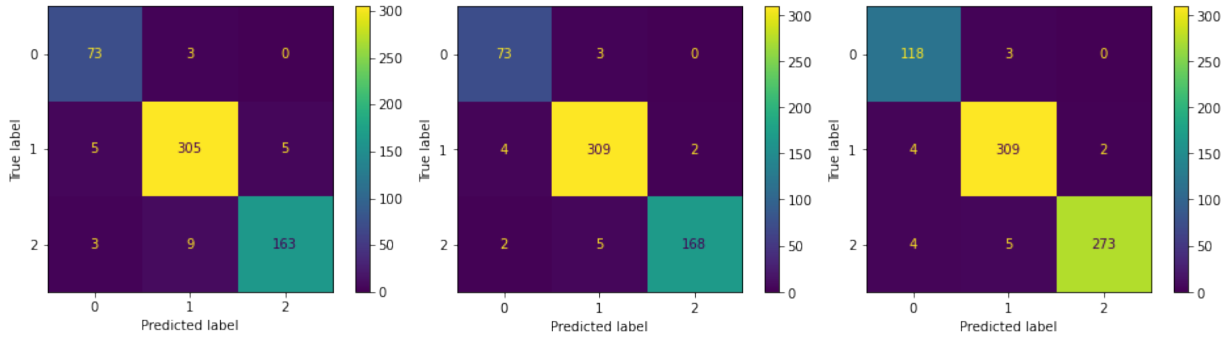
Similarly, in Hanover the 9 recoverable regulators shown in Table 5.9 were recovered (cf. Figure 5.9g and 5.9h, the true positives increased from 541 to 550). 159 new regulators were predicted from arms with missing data, accounting for 29.4% of the original dataset (cf. Figure 5.9h and 5.9i), of which all were correct (100% accuracy).



(a) Champ: initial. *(b)* Champ: corrected rules. *(c)* Champ: corrected and inferred.



(d) Chic: initial *(e)* Chic: corrected rules. *(f)* Chic: corrected and inferred.



(g) Han: initial. *(h)* Han: corrected rules. *(i)* Han: corrected and inferred.

Figure 5.9.: Confusion matrices before checking for consistency the predicted labels using domain knowledge rules (a, d, g), after recovering incorrect predictions with the usage of domain knowledge rules (b, e, h), and after both correcting incorrect predicted labels and inferring regulation for context arms with no available trajectory data (c, f, i).

Therefore, by applying domain knowledge rules, there is a gain in accuracy between 1% and 3%, but more importantly, accurate predictions (99% in Champaign, 98% in Chicago and 100% in Hanover) can be made to a number of arms with incomplete data that corresponds to 29%-49% of the original data. Furthermore, the FPR of the class with the highest FPR decreases between 15.6% to 60% (15.6% in Chicago, 60% in Champaign and 58.3% in Hanover), validating this thesis' proposal to use domain knowledge rules for both recovering misclassified regulators and for predicting regulators for arms with no trajectory data.

5.4. Discussion

The main findings of this chapter are the following:

1. The TRR method proposed in this chapter (hybrid all static model), which combines data from trajectories and OSM and makes use of information of context intersection arms, can provide accurate predictions for rule sets consisting of UN, SS, PS, and TS: 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 the classification: all-arm classification models outperformed single-arm models.

3. The sampling rate can affect the performance of the classification. A low sampling rate 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 1-2% when the sampling interval was doubled from 2 s to 4 s.

4. The GB classifier performs as well or better (increase in F1-score between 1%-4%) than RF for all experiments (cf. all-arm models in Table 5.2).

5. 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%-3%. Therefore, the exclusion of curved trajectories has a positive effect on the classification performance.

6. The optimal number of trajectories per intersection arm for the computation of classification features found to be five straight trajectories.

7. Eliminating the number of trajectories that cross the intersections arms to a certain number, for the computation of the classification features, negatively affects the classification performance. However, with only three straight trajectories per intersection arm, the 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, the accuracy is equal to or greater than 90% (90% in Chicago and 92% in Champaign and Hanover).

8. Domain knowledge rules in general can help recover incorrect predictions in cases where there are predictions available from the context arms and there is inconsistency in the predicted labels, expressed (and detected) as a significant difference in the predicted probabilities of the labels (threshold value for probability difference used: 0.14). If all regulations of the same intersection are predicted with high probability (cases marked in green in the tables analysing the misclassifications), then unless there is a majority arm label agreement, no recovering action can be triggered. In the three datasets examined, the majority of intersection arms has only 1 or no context arms. In total, of the 63 incorrectly predicted arms in the three datasets, only 3 arms have more than 1 context arm (2 context arms), where each arm can be examined by comparing its predicted label with the two predicted labels of its context arms. The more context arms are available, the more potential for domain

rules to be used to recover incorrect predictions. Furthermore, domain rules cannot help in cases where all context arms are misclassified, but the predicted labels do not contradict the rules (cases marked in white in the relevant tables), for example, all arms are predicted as TS, when in fact they are SS, since all-way-stop controlled is a valid regulation for an intersection. By applying domain knowledge rules, there is a gain in accuracy between 1% and 3%, but more importantly, accurate predictions can be made on a number of arms with incomplete data corresponding to 29%-49% of the original data. Interestingly, the (incomplete) arms predicted solely based on the information of context arms are predicted with high accuracy: 99% in Champaign, 98% in Chicago and 100% in Hanover. In addition, the FPR of the class with the highest FPR decreases between 15.6% to 60% (15.6% in Chicago, 60% in Champaign and 58.3% in Hanover), validating the proposal of this thesis to use domain knowledge rules to both recover incorrect predicted regulation labels and to predict regulators for arms with no available trajectory data.

An interesting direction to extend this study would be on how traffic regulations can be predicted with high accuracy under limited labeled data. As discussed in Section 4.2, labeling arms with regulations for a TRR learning task, is a manual and costly process, and although there are many trajectory datasets that one can use for TRR, the need for labeled data makes the latter unsuitable for this purpose. One idea is to explore unsupervised methods, such as clustering, where the data needs not to be labeled. A second idea is to explore semi-supervised methods, such as self-training (Van Engelen and Hoos, 2020), where only limited labelled data is 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 predicting target labels in a city Y, assuming no labeled data is available from the latter city. These ideas are explored in Chapter 6.

5.5. Summary

In this chapter, a new TRR method (hybrid all-static) was proposed which combines data from trajectories and OSM and uses information from context intersection arms. The methodology was tested in three datasets containing 3-way and 4-way intersections, controlled by UN, SS, TS (Champaign, Chicago) and UN, PS, TS (Hanover). Curved trajectories found to have a negative effect on the classification accuracy by 1%-3%, therefore extracting classification features only from straight trajectories can positively affect the classification performance. The minimum optimal number of trajectories per intersection arm was found to be five straight trajectories. Furthermore, limiting the number of trajectories to a certain number negatively affects the classification performance. However, with only three straight trajectories per intersection arm, the 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, the accuracy is equal to or greater than 90% (90% in Chicago and 92% in Champaign and Hanover). Finally, by applying a set of domain knowledge rules to the predicted labels on an intersection level, we were able to recover misclassified regulators and predict labels from arms with no data. By applying domain knowledge rules, there is a gain in accuracy of between 1% and 3%, but more importantly, accurate predictions can be made on a number of arms with no available trajectory data corresponding to 29%-49% of the original data. Interestingly, the regulators from (incomplete) arms predicted solely based on the context arm information are predicted with high accuracy 99% in Champaign, 98% in Chicago and 100% in Hanover. The proposed model (using at least five trajectories per intersection arm) after recovering potentially identified incorrect predictions by applying domain knowledge rules achieved an accuracy of 97% in Champaign and Hanover, and 95% in Chicago (F1-score: 0.97, 0.98 and 0.94 respectively).

5.6. Acknowledgements

The methodology described in this chapter, as well as the other experiments proposed, tested and presented here, are published in the ISPRS International Journal of Geo-Information under the title *Recognition of Intersection Traffic Regulations From Crowdsourced Data* (Zourlidou et al., 2023). The author of this thesis is the principal investigator and first author of the article, as well as the creator of all figures.

6. TRR with Sparsely Labeled and Stream Data

6.1. Introduction

In this chapter, the default TRR model is examined under different scenarios of availability of labeled data. Starting with the scenario of having available only trajectories and the free OSM, i.e. no regulation labels for training a classifier, it is investigated whether the use of *clustering* and some mean statistics of the data instances clustered in the same cluster can lead to an accurate classification of intersection arms (Section 6.2). Section 6.3.1 explores the scenario where a small amount of labeled data is available and both labeled and unlabeled data are used in the training process (*self-training*). The same scenario is also explored in Section 6.3.2 by selecting a small set of data to be labeled with a process that queries data instances for which the classifier is less confident about their predicted target labels (*active learning*). Section 6.3.3 tests the idea of using clustering for selecting the data instances to be labeled (*cluster-then-label*). A comparison of all classification methods is presented in Section 6.3.4. In addition, in Section 6.4, it is investigated whether learning is transferable between cities (*learning transferability*), assuming that there is no labeled data from a city A whose regulations need to be predicted, but do exist labeled data from another city B. We then examine whether training a classifier with data from city B can provide accurate predictions for the regulators of city A. Finally, Section 6.5 tests the default TRR model in incremental learning settings.

6.2. TRR with Clustering

Clustering for grouping data instances in certain categories is not common, as after clustering one needs to match clusters to target categories. However, in the context of the TRR problem, Hu et al. (2015) have tested the idea of using clustering for grouping the intersection arms in three categories (UN, SS, TS), achieving classification accuracy of 80%. More specifically, after applying clustering, they used the mean statistics of the crossing speed and the duration of the last stop episode within the clustered dataset instances of each cluster to match the three clusters in regulation categories. The hypothesis of this idea is that if clustering is done successfully, then the target labels can be recovered by the characteristic mean values of the above mentioned features. As authors explain (Hu et al., 2015, p. 23) “common sense tells us, for example, if a car does stop at a traffic regulator, then a red light would be the regulator that causes the longest wait; and when considering the lowest passing speeds a car demonstrates when crossing different intersections, at the uncontrolled ones we would observe the highest speed compared to traffic lights or stop signs”.

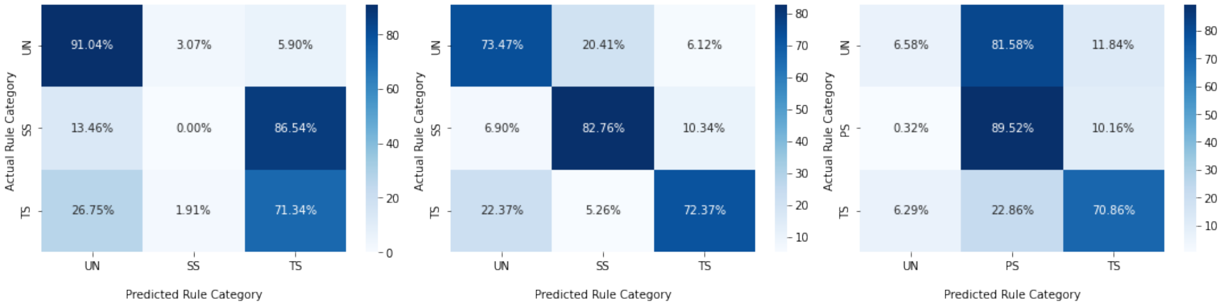
We adopted this idea and tested it on the three datasets. The K-means algorithm (Section 2.3.3.1) was used to cluster the data into three categories. After clustering, the mean values of the two features were calculated. Table 6.1 shows these values for each cluster. For the Champaign and Chicago dataset, we assumed that the cluster having the largest average duration of the last stopping episode corresponds to traffic light category (cluster

Table 6.1.: Mean values of the duration of the last stop episode and of the average crossing speed, calculated from the data assigned to each cluster.

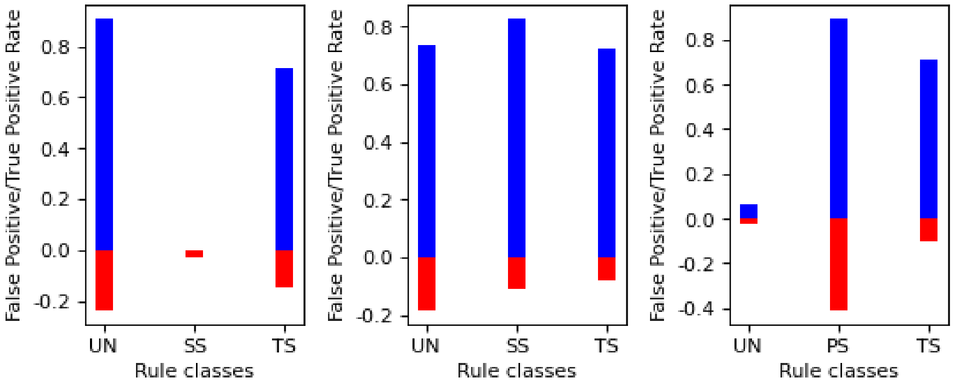
Dataset	Classification Feature	Cluster A	Cluster B	Cluster C
Champaign	average speed (kmh) duration of last stop episode (s)	49.6 1.75	29.61 12.09	22.87 6.16
Chicago	average speed (kmh) duration of last stop episode (s)	23.9 6.6	40.4 2.5	29.2 11.7
Hanover	average speed (kmh) duration of last stop episode (s)	32.67 20.19	36.88 3.21	25.48 24.46

Table 6.2.: Classification performance after matching clusters to regulator labels.

Dataset	Recall	Precision	F-score	Accuracy
Champaign	0.75	0.79	0.77	0.79
Chicago	0.77	0.75	0.75	0.75
Hanover	0.68	0.73	0.69	0.73

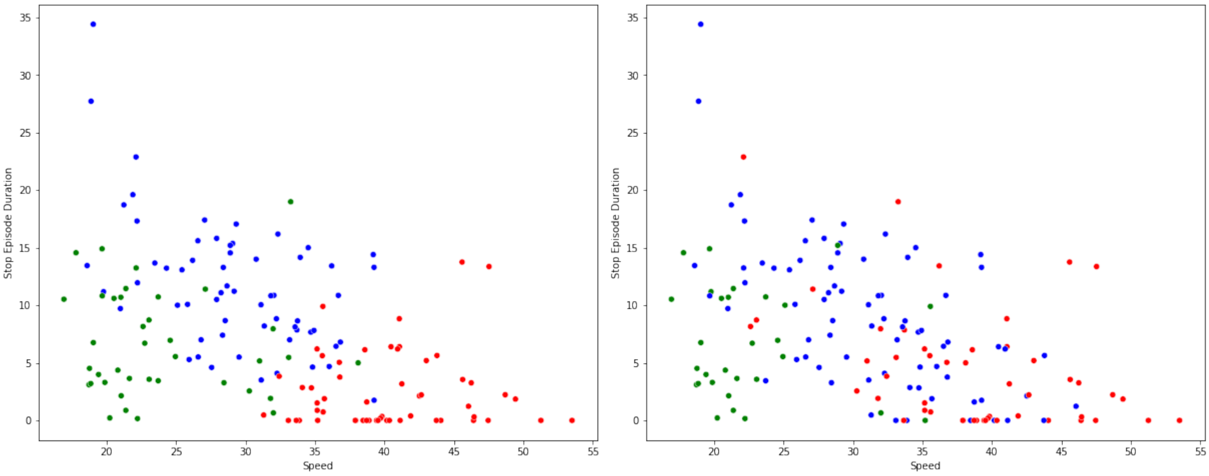


(a) Champaign. (b) Chicago. (c) Hanover.

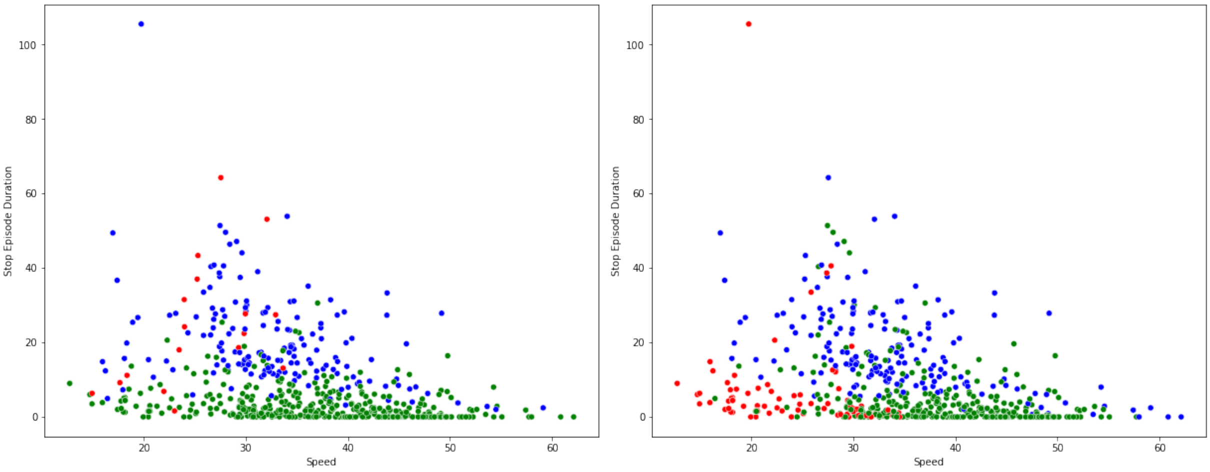


(d) Champaign. (e) Chicago. (f) Hanover.

Figure 6.1.: Confusion matrices and false/true positive rates for the three datasets after applying K-means clustering.



(a) Chicago: predicted labels. *(b)* Chicago: actual labels.



(c) Hanover: predicted labels. *(d)* Hanover: actual labels.

Figure 6.2.: Predicted labels vs. actual labels. The x-axis corresponds to the average crossing speed and y-axis to the duration of last stopping episode. In Chicago, green corresponds to SS, blue corresponds to TS and red to UN. In Hanover, green corresponds to PS, blue to TS and red to UN.

B for Champaign and cluster C for Chicago) and the cluster with the largest average speed corresponds to uncontrolled intersections[1], which is cluster A for Champaign and cluster B for Chicago. The remaining cluster is then assigned to the stop sign. For the Hanover dataset, similar knowledge matching rules were applied. The priority sign was assigned to the cluster with the highest speed (cluster B), and since clusters A and C have very similar values for stopping time, the cluster from these two clusters with the highest speed was assigned to the traffic signal category (cluster A) and the remaining cluster C was assigned to the uncontrolled arms[2]. The performance of the classification after these two

---

[1] In Champaign and Chicago, as it will be discussed in Section 6.4, the vast majority of uncontrolled arms coexist with arms with stop signs at the same intersection (cf. Table 6.4), so vehicles passing through uncontrolled arms, cross the intersection at high speed.

[2] In Hanover, as it will be discussed in Section 6.4, the uncontrolled arms coexist in the same intersection only with the uncontrolled arms (cf. Table 6.4), which means that a vehicle passing through such intersections will have to slow down/stop to check/yield for traffic coming from the right, so its speed will be lower than at a priority controlled arm or a traffic light.

steps (clustering and cluster-label matching) is reported in Table 6.2 (a detailed report with per class performance is given in Table A.5 in the Appendix on page 153). Indeed, the accuracy is close to the accuracy of 0.8 reported in Hu et al. (2015). However, the confusion matrices (Figure 6.1) show that in the two largest datasets, the class SS in Champaign and UN in Hanover are largely, if not completely misclassified. The difference between the predicted labels and the actual ones is also illustrated in Figure 6.2. In Chicago the three classes are almost equally well distinguished (cf. Figure 6.2a with Figure 6.2b). However, in Hanover the class with the fewer examples (UN) is largely misclassified (cf. Figure 6.2c with Figure 6.2d). Testing other clustering techniques, such as DBSCAN and spectral clustering, did not yield better results. Therefore, we conclude that not having labeled data and using clustering for TRR, can only provide poor classification performance. In the next section, we investigate whether using small amounts of labeled data, all regulatory categories can be accurately identified.

6.3. TRR with Self-Training, Active Learning and Cluster-then-Label

6.3.1. TRR with Self-Training: Using Labeled and Unlabeled Data

In the case where there is a small amount of labeled data available (in our context, intersection arms for which both trajectory data and regulation labels are available) and an additional large amount of unlabeled data (in our context, intersection arms crossed by trajectories), self-training approaches can be used to exploit *both* labeled and unlabeled available data. Here we test the Algorithm 5, explained in 2.3.4.1, using as confidence threshold 0.95 and as stopping criterion the labeling of all unlabeled data. Only the two largest datasets were used for the experiments, as the Chicago dataset was too small for the requirements of the methodology (a small amount of labeled data is needed plus a bigger amount of unlabeled data). The *SelfTrainingClassifier* function, provided by the Python library *sklearn.semi_supervised*, was used for the implementation.

The settings of the experiments are shown in Figure 6.3. Here, we split the dataset into training and testing datasets, so that the classification accuracy of all experiments (baseline vs. self-training) refers to performance measured on the same dataset. More specifically, experiments were performed for different amounts of labeled data starting with 2% of the total available data set and gradually increasing the amount of training data in 2% increments up to 42%, as illustrated in Figure 6.4. The 2% of data corresponds to 12 intersection arms for the Champaign dataset and 11 for the Hanover dataset. As Figure 6.4 illustrates, each successive experiment uses the same training examples used in the previous experiment plus 2% new training examples, randomly selected from the pool of unused training examples (blue in Figure 6.3). Once new labeled examples are sampled for training (all labeled data are illustrated in green in Figure 6.3), the remaining (training) data are masked as unlabeled data (depicted in blue). The classification accuracy of the TRR model using self-training is compared to the performance of the same TRR model trained using the same labeled

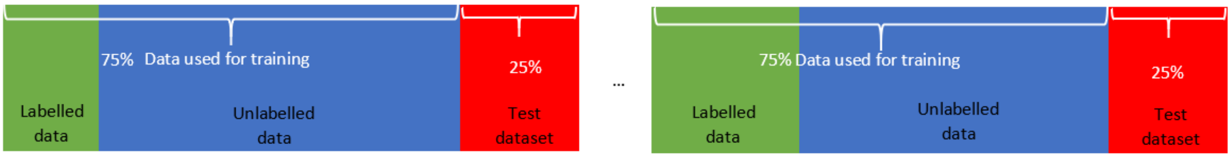
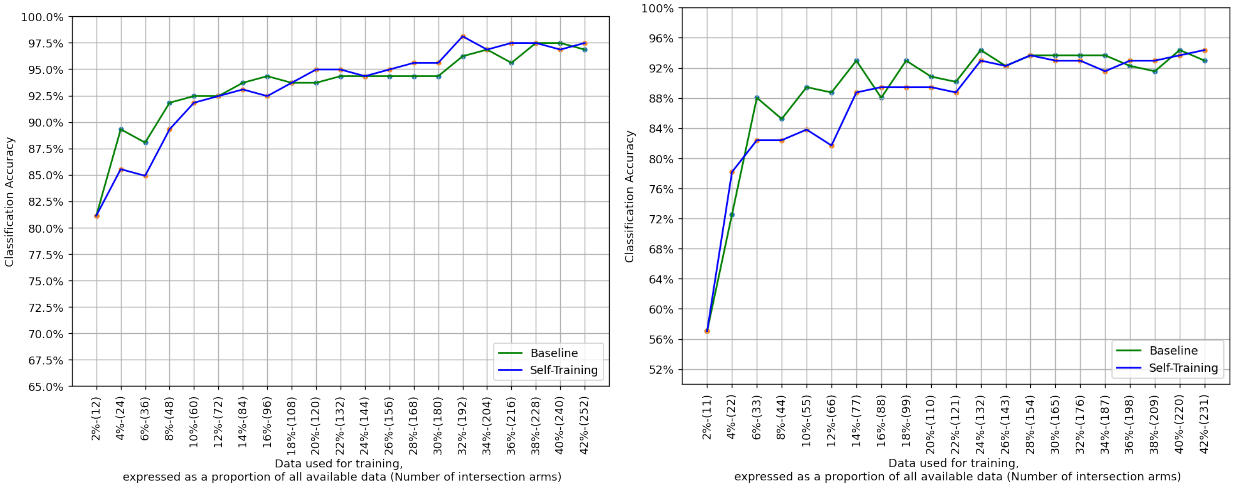


Figure 6.3.: Settings of self-training experiments using different sizes of labeled/unlabeled data.



(a) Champaign. *(b)* Hanover.

Figure 6.4.: Classification performance using self-training under different amounts of labeled/unlabeled training data. The percentages of data used as training data of x-axis refer to labeled data.

examples used in the corresponding self-training experiment (we call it *baseline* from now on). In other words, the baseline model is trained with the same labeled examples as the self-training model, but the difference between them is that the latter uses for training in addition to labeled data, unlabeled data as well. The performance of both models is tested in the same test dataset, depicted in red in Figure 6.3. To sum up the experiment settings, we split the dataset into training and testing data; the baseline model is trained on the labeled data (green) and tested on the testing dataset (red), and the self-training model is trained on the labeled and unlabeled data (green and blue) and tested on the testing dataset (red).

Figure 6.4 shows the classification results of self-training in the two datasets. In the Champaign dataset (Figure 6.9a), the classifier using self-training with labeled data that corresponds from 2% of the total available data, until 18% (and additionally 73% to 57% respectively unlabeled data), performs worst than the baseline model. After using above 120 labeled examples (it corresponds to 20% of the total amount of data), the self-training model starts to outperform slightly the baseline model. In the Hanover dataset (Figure 6.4b), for 4% of labeled data the self-training model outperforms the baseline model but then starts performing worst than the later until 26% of the data. After that, both models perform similarly. In both datasets, after around 144 labeled examples the two models perform similarly. Before that number, the self-training model fails to outperform the baseline model almost in all experiments. Therefore, we conclude that using self-training under the datasets characteristics (e.g., number of examples) and the experimental settings (percentages of labeled-unlabeled examples) we tested, in the context of the TRR problem, no improvement in performance is achieved compared to the baseline model.

6.3.2. TRR with Active Learning

In this section, we investigate whether querying the labels of intersection arms for which the classifier makes less confident predictions, as explained in Section 2.3.5, can yield better performance than randomly acquiring labels. Or in other words, whether we can use active learning to reduce the number of labeled examples needed to train a classifier that can

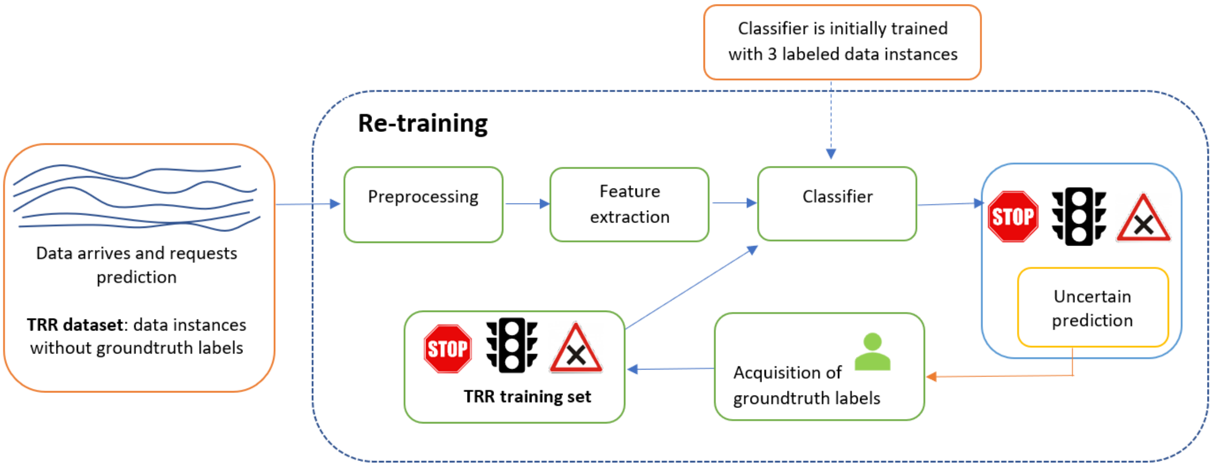

Classifier is initially trained with 3 labeled data instances
Re-training
Data arrives and requests prediction
TRR dataset: data instances without groundtruth labels
Preprocessing
Feature extraction
Classifier
STOP
Uncertain prediction
STOP
TRR training set
Acquisition of groundtruth labels

Figure 6.5.: Active learning framework for traffic regulation recognition.

generalize well on unseen data (test dataset). Figure 6.5 illustrates an active learning framework for TRR. For the implementation, we used the function *models.ActiveLearner* from the Python *modAL* framework (Danka and Horvath, 2018). The active learning algorithm has been described in Section 2.3.5 (Algorithm 7).

The settings for the experiments are shown in Figure 6.6. Similar to the experiments for testing self-training, the experiments here were run for different amounts of labeled data, starting with 2% of the total available data set and gradually increasing the amount of training data in 2% increments up to 42%, as shown in Figure 6.7. The 2% of data corresponds to 12 intersection arms for the Champaign dataset and 11 for the Hanover dataset. Each successive experiment uses the same training examples used in the previous experiment plus 2% of new training examples selected randomly from the pool of unused training examples for the baseline model and selected using pool-based uncertainty sampling for the active learning model, as explained in Section 2.3.5. The classification accuracy of the TRR model with active learning is compared to the performance of the baseline model trained with the same amount of labeled examples. In order to compare the classification performance between self-training and active learning on exactly the same test dataset, we used the same training/test settings here as for self-training. Therefore, the baseline model was trained on *randomly selected* labeled data (green in Figure 6.3) and tested on the test dataset (red), and the active learning model was trained on the *actively labeled* training data (dark green in Figure 6.6) and tested on the test dataset (red, which is the same dataset used for testing the baseline and self-training models).

Figure 6.7 shows the classification results of active learning in the two datasets. In both datasets, the model trained with active learning outperforms the baseline model. In particular, in the Champaign dataset (Figure 6.7a), the classification accuracy reaches 95% with

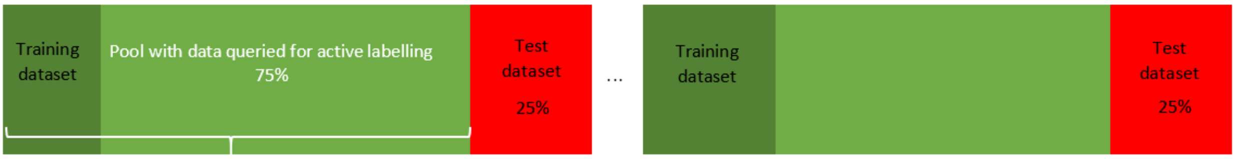
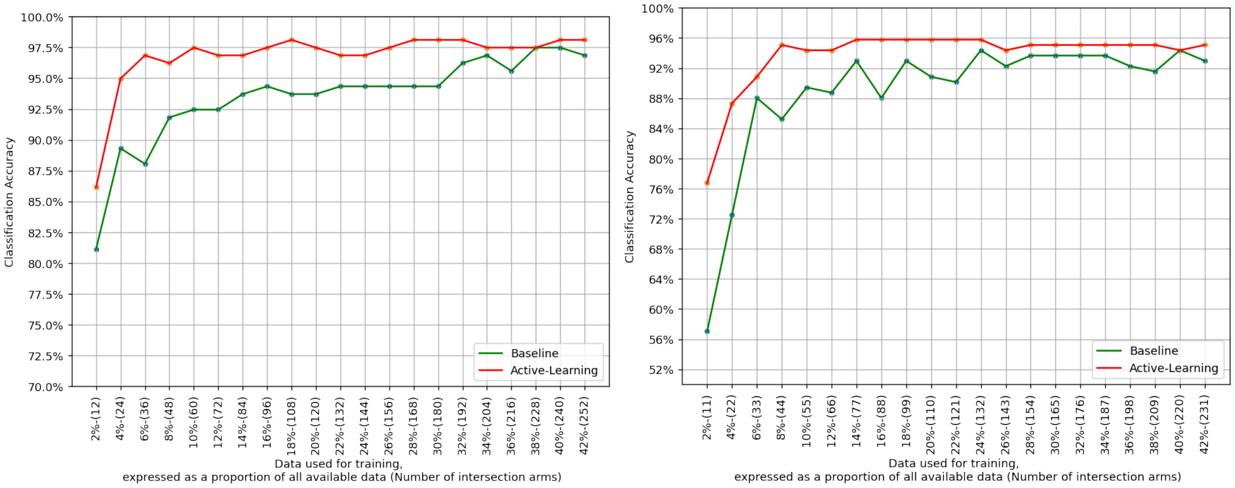


Figure 6.6.: Settings of active-learning experiments with different sizes of actively labeled training data. Successive experiments query a larger amount of labels from a hypothetical human domain expert (annotator), as explained in Section 2.3.5.



(a) Champaign. *(b)* Hanover.

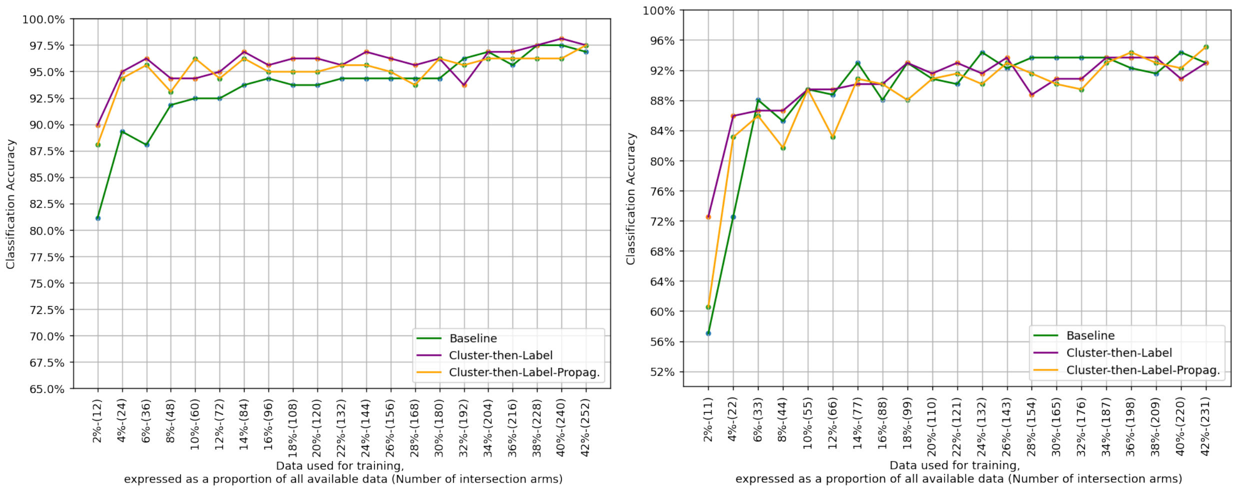
Figure 6.7.: Classification performance using active learning under different amounts of (actively) labeled training data.

only 24 actively labeled data instances (4%). The baseline model achieves similar performance with 132 data instances (22%). Moreover, the active learning model starts converging at 36 data instances (accuracy 96%-97%), while the baseline model converges at an accuracy of over 95% at 192 instances (32%). In the Hanover dataset (Figure 6.7b), the active learning model achieves 88% accuracy with only 22 instances (4%). The corresponding accuracy of the baseline model is 72%. The active learning model starts converging with 44 examples (8%) (accuracy 95%), while the baseline model needs 132 examples (24%) to achieve an accuracy close to that value. Also for both datasets, with just 44 examples which corresponds for example to 11 four-way intersections, one can expect an accuracy of over 95%.

Thereof, it can be concluded that using active learning under the dataset characteristics (e.g., number of examples) and experimental settings (percentage of labeled data) we tested in the context of the TRR problem, can lead to an increase in classification performance compared to the baseline model when trained with the same amount of training data.

6.3.3. TRR with the Cluster-then-Label Algorithm

Here we test Algorithm 6 (cluster-then-label), which is explained in Section 2.3.4.2 on page 59. The classification performance is shown in Figure 6.8. Briefly, given a set of unlabeled data and a number k corresponding to the number of data instances to be manually labeled, we cluster the data with the K-means algorithm into k clusters (k corresponds to the number of arms of the x-axis of Figure 6.8). Then, for each cluster, the data instance x closest to the centroid of the cluster is labeled. After that, the XGBoost classifier is trained with the examples labeled in the previous step and is ready for predicting unlabeled data. A variation of this algorithm is to propagate the label of the x data instance to p percent of the data instances that belong to the same cluster with x, and repeat this process for all k clusters. We refer to this classification process as *cluster-then-label-propagation*. Both cluster-then-label and cluster-then-label-propagation learning were tested and compared to the baseline model. Label propagation in the second model was implemented by propagating the label of each cluster centroid x to all data instances of the same cluster with x (parameter p of Algorithm 6 was set to 100%).



(a) Champaign. *(b)* Hanover.

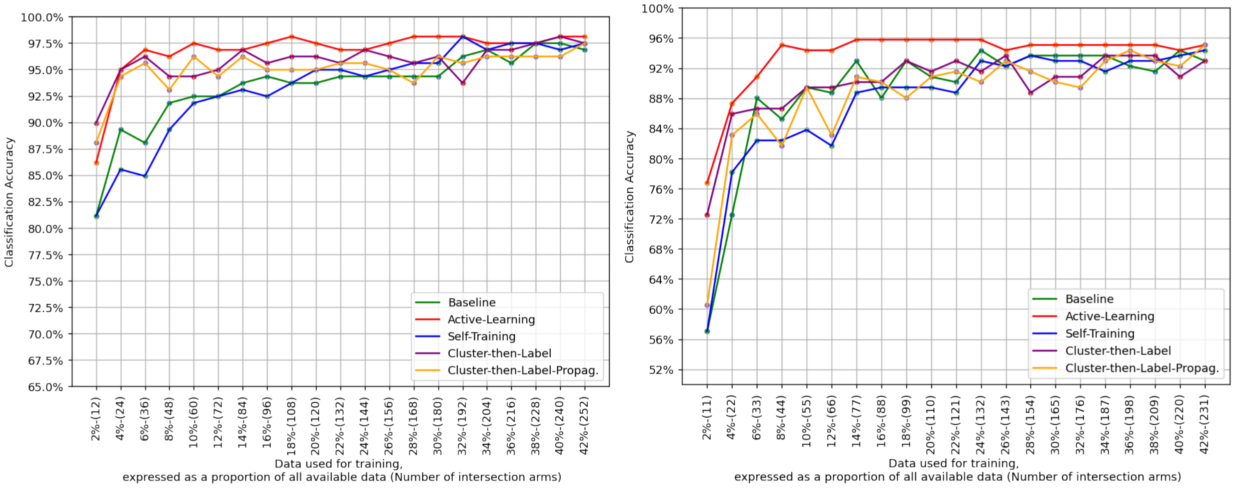
Figure 6.8.: Classification performance of cluster-then-label and cluster-then-label-propagation under different amounts of labeled training data.

In both datasets, the cluster-then-label learning performs better *without* label propagation than with label propagation, with very few exceptions and very little difference in performance when this is the case (e.g., for 12% of the labeled data in Champaign). More specifically, the cluster-then-label learning in Champaign outperforms the baseline model in all experiments from 2% to 30%, and after this value the two models are very close in terms of accuracy. The fewer labeled examples there are, the greater the difference in performance between the baseline model and the cluster-then-label classification. However, for the Hanover dataset the same observations do not hold. Although cluster-then-label performs better than cluster-then-label-propagation in all experiments up to 40% of the labeled data, a significant difference between the two models is only seen for 2% and 4% of the labeled data. After these values, both models perform similarly.

Therefore, it is difficult to make a general statement about the performance of cluster-then-label learning compared to the baseline model. It can only be stated that the use of cluster-then-label learning is more efficient than the baseline model under the characteristics of the datasets (e.g.., number of examples) and experimental settings (percentage of labeled data) that we tested in the context of the TRR problem for very small amounts of labeled data such as 11 or 21 data instances (cf. 2% and 4% in Hanover and Champaign, respectively), and it cannot guarantee further improvement for larger values of labeled data, since such improvement was observed only in Champaign and not in Hanover.

6.3.4. Comparison of All Tested Methods

The performance of all learning methods tested and discussed in this Section 6.3 is shown in Figure 6.9. Active learning clearly outperforms all learning methods including the baseline model. Moreover, it can be seen that all methods, after being trained with 40% of the available data (240 arms in Champaign and 220 arms in Hanover), end up predicting the intersection rules with similar accuracy. From a practical point of view, however, the possibility of using active learning to achieve high prediction accuracy with less training data remains interesting: by selecting the intersection arms to be manually labeled using a sampling strategy from those suggested by the active learning framework, the number of



(a) Champaign. *(b)* Hanover.

Figure 6.9.: Classification performance of cluster-then-label, cluster-then-label-propagation and baseline model under different amounts of labeled training data.

training data required for accurate predictions is significantly reduced. For both datasets, we found that with just 44 examples which corresponds for example to 11 four-way intersections, one can expect an accuracy of over 95%. Compared to the number of data instances of the baseline model that achieves the same accuracy (132 in both datasets), active learning reduces the number of training data by 66.7%.

6.4. Learning Transferability: Training on City A and Predicting on City B

This section examines the scenario of predicting regulations of a city B, using a classifier trained with data from another city, say city A. Such a scenario is illustrated in Figure 6.10.

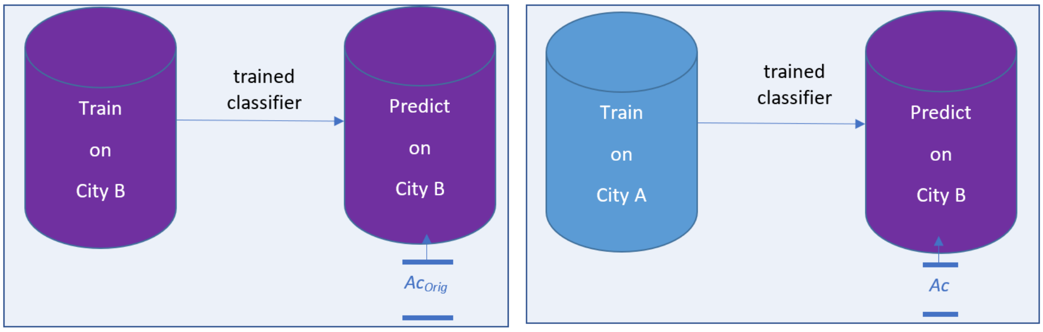
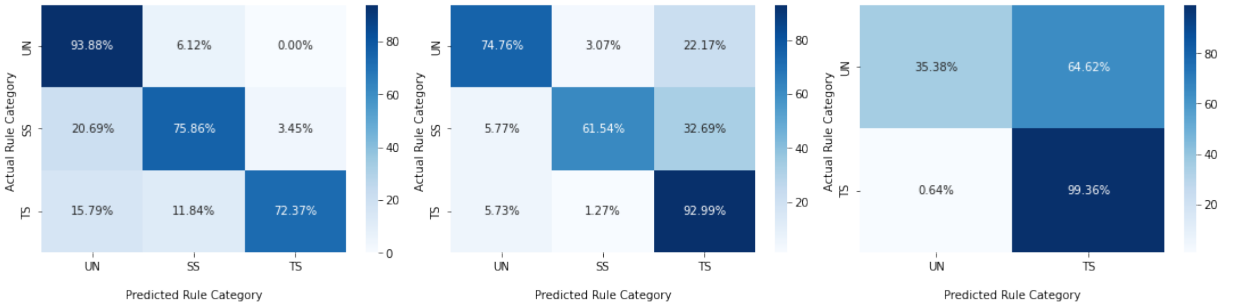


Figure 6.10.: Experiments on transferability of learning are applied as illustrated in the right figure. A TRR classifier is trained on data from a city A and then applied to data from a city B. The classification accuracy is denoted as Ac. If training and predicting are performed on the same dataset, as illustrated in the left figure, then the accuracy is denoted as Ac_{orig}*.*

Here the assumption is that learning can be transferable within cities. To test this hypothesis, six experiments were conducted within the three cities of Champaign, Chicago, and Hanover. The settings of the experiments and the *Ac* classification accuracy of the experiments are presented in Table 6.3. The classification accuracy of each experiment is compared

Table 6.3.: Classification results of the six experiments on the transferability of learning between cities. Ac: accuracy on the Predict dataset when training is done with data from the city given in the Training dataset column; $\Delta_{Ac} = Ac_{Orig} - Ac$, where Ac_{Orig} is the accuracy when training is done in the same city as the Predict dataset (Ac_{Orig} is given in the lower block of rows of the table). The best accuracy per experiment across the different TRR methods is indicated in bold.

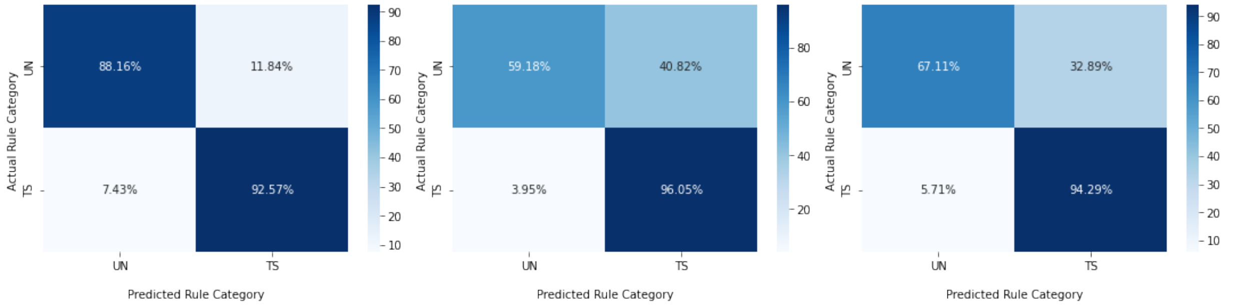
				Static (all-arm)		Dynamic (one-arm)		Dynamic (all-arm)		Hybrid (one-arm)		Hybrid (all-static)	
Exp.	Training Dataset	Predict Dataset	Regulators	Ac	Δ_{Ac}	Ac	Δ_{Ac}	Ac	Δ_{Ac}	Ac	Δ_{Ac}	Ac	Δ_{Ac}
1	Champ.	Chic.	UN, SS, TS	0.63	0.18	0.75	0.05	0.73	0.12	0.73	0.09	**0.80**	0.11
2	Chic.	Champ.	UN, SS, TS	0.53	0.30	**0.83**	0.10	0.82	0.13	**0.83**	0.11	0.78	0.16
3	Han.	Champ.	UN, TS	0.45	0.48	0.35	0.60	0.33	0.62	0.41	0.54	**0.53**	0.43
4	Champ.	Han.	UN, TS	0.77	0.20	**0.92**	0.05	0.90	0.07	**0.92**	0.06	0.91	0.07
5	Han.	Chic.	UN, TS	0.73	0.22	0.66	0.21	0.62	0.29	0.65	0.22	**0.82**	0.12
6	Chic.	Han.	UN, TS	0.78	0.19	0.84	0.13	0.82	0.15	0.85	0.13	**0.86**	0.12
				Ac_{Orig}		Ac_{Orig}		Ac_{Orig}		Ac_{Orig}		Ac_{Orig}	
	Chic.	Chic.	UN, SS, TS	0.81		0.80		0.85		0.82		0.91	
	Champ.	Champ.	UN, SS, TS	0.83		0.93		0.95		0.94		0.94	
	Chic.	Chic.	UN, TS	0.95		0.87		0.91		0.87		0.94	
	Champ.	Champ.	UN, TS	0.93		0.95		0.95		0.95		0.96	
	Han.	Han.	UN, TS	0.97		0.97		0.97		0.98		0.98	



(a) Exp1: Train in Champ. and predict in Chic. (acc: 0.80).

(b) Exp2: Train in Chic. and predict in Champ. (acc: 0.78).

(c) Exp3: Train in Han. and predict in Champ. (acc: 0.53).



(d) Exp4: Train in Champ. and predict in Han. (acc: 0.91).

(e) Exp5: Train in Han. and predict in Chic. (acc: 0.82).

(f) Exp6: Train in Chic. and predict in Han. (acc: 0.86).

Figure 6.11.: Confusion matrices of the hybrid all-static model for the six experiments.

to the corresponding Ac_{orig} accuracy if the training was done with data from the same city as the test, as depicted in Figure 6.10 (left). The difference between Ac_{orig} and Ac is given in Δ_{Ac} column in Table 6.3. Comparing the classification results of the six experiments, the following observations can be made, that lead also to critical questions:

1. The hybrid all-static method scored better than the other TRR methods in four of the six experiments. Figure 6.11 shows the confusion matrices of the hybrid all-static model for the six experiments. This finding contrasts with the finding discussed in Chapter 5 (cf. Table 5.2), where the hybrid all-static models scored better than the other models in *all* three datasets. *The question is why in two experiments the hybrid all-static did not work as expected.*

2. The dynamic all-arm model, in *all* six experiments performed slightly worse than the dynamic one-arm model. This finding contrasts with the finding discussed in Chapter 5 (cf. Table 5.2), where the dynamic all-arm models scored slightly better than the dynamic one-arm models in all three datasets. *The question is why in all experiments the dynamic all-arm models performed worse than the dynamic one-arm models.*

3. The static model performed worse than the hybrid all-static in all experiments, which is consistent with the finding discussed in Chapter 5 (cf. Table 5.2, the hybrid all-static scored better than the static all-arm models in all three datasets).

4. In five of six experiments, the best accuracy in the TRR methods tested is above 0.80. Only in Exp3, the accuracy is very low (0.53 in hybrid all-static). Also, in Exp3 and Exp5 the Δ_{Ac} of the dynamic models (one-arm and all-arm) as well as that of the hybrid one-arm model has a very high value: e.g., for the hybrid one-arm model in Exp3, $\Delta_{Ac} = 0.54$ and in Exp5, $\Delta_{Ac} = 0.22$. Moreover, the difference in accuracy between Exp3 and Exp4, and between Exp5 and Exp6 is remarkable. For example, when training is done in Hanover and predicting in Champaign, we observe very poor performance in the dynamic and hybrid models. When training is done in Champaign and predicting in Hanover the performance is high. *The question is how to explain such a large value of Δ_{Ac} and why the performance within the same cities differs so much when the cities swap training and predicting positions.*

5. For the six experiments, the original trajectory datasets were used, which, as explained previously, have different sampling rates. In Section 5.3.2, it was shown that increasing the GPS sampling interval has a negative effect on the classification accuracy. However, here the training is done not only with data from different geographical locations but also with different characteristics in terms of their sampling interval. *Therefore, it would be interesting to examine whether having the training and test datasets similar sampling rates, would positively affect the classification performance.* To test this hypothesis, we will (under)sample the Champaign and Hanover datasets as follows: for Exp1 and Exp2 undesample Champaign to ≈ 4 s (to be closer to Chicago's sampling interval), for Exp3 and Exp4 undersample Champaign to ≈ 2 s (to be closer to Hanover's sampling interval) and for Exp5 and Exp6 undersample Hanover to ≈ 4 s (to be closer to Chicago's sampling interval).

Exploring the questions raised by the above observations, the following explanations can be given:

1. (Explanation of observation 1) The hybrid all-static method scored better than the other TRR methods in four of the six experiments. In Exp2 the dynamic one-arm and the hybrid-one-arm models scored better than the hybrid all-static model (0.83 vs 0.78). A similar case is Exp4, where the dynamic one-arm and hybrid-one arm scored an accuracy 0.92 and the hybrid all-arm model scored 0.91. Here we will attempt to

explain the 0.05 difference in accuracy of Exp2. In Exp2, we can see that the static model performs poorly (accuracy 0.53) compared to the other TRR models (dynamic one-arm: 0.83, dynamic all-arm: 0.82, hybrid one-arm: 0.83 and hybrid all-arm: 0.78). The large difference in accuracy in the static model (Δ_{Ac}=0.30), could be explained by the fact that the Chicago dataset geographically covers an area of 3.9 km x 2.5 km, while the Champaign dataset covers an area of 12.2 km x 13 km (cf. Figure 4.1). When the classifier is trained on the Chicago dataset and then tries to predict regulators in the Champaign dataset, the map-based individualities of the larger test dataset, as reflected in the static features from all context arms, may not be able to be captured by a classifier that has been trained on a much smaller dataset (static all-arm model accuracy=0.53). This observation may also explain why the hybrid all-static (accuracy 0.78) that uses all the static features from the context arms of an intersection has 5% worse performance than the dynamic one-arm model and the hybrid one-arm model (accuracy 0.83). Therefore, in transferability learning settings, static information from context arms can generally benefit TRR models unless the training dataset is much smaller than the test dataset, as in Exp2, a finding that is consistent with the finding discussed in Chapter 5 (cf. Table 5.2, the hybrid all-static and hybrid all-arm models scored better than the hybrid one-arm models on all three datasets).

2. (Explanation of observation 2) The dynamic all-arm model, in *all* six experiments performed slightly worse than the dynamic one-arm model. In order to investigate in what ways the dynamic information from context arms differs within the training the test datasets, we investigated what kind of dynamic information is found within an intersection, i.e., how many arms are crossed by at least five straight trajectories and by what regulations they are controlled, within each intersection of the datasets. This information for all intersections containing arms that are sampled with at least five straight trajectories for the three datasets is presented in Table 6.5. For example, in Champaign there are 170 three-way and 197 four-way intersections that are sampled with at least one arm crossed by five straight trajectories. From the 170 three-way intersections, 115 intersections (5th column) have two UN arms (UN UN) that are sampled by at least five straight trajectories each, 3 intersections have two arms regulated by UN and SS (7th column) and are sampled accordingly, etc. As we can see in Table 6.5, the dynamic information between datasets differs in both qualitative and quantitative aspects. Between Champaign and Chicago the difference is smaller, but still one can see that in training settings, a classifier trained on the Champaign dataset will use examples for recognising UN regulated arms, which in the dynamic all-arm settings will contain information from context arms that are regulated as UN SS (see 4th and 22nd column in Table 6.3), UN (5th and 23rd columns), SS (7th column) and individual arm UN related information (no information from context arms available, see 11th and 26th columns). For the same training task for UN recognition in the Chicago dataset, the classifier will use examples which contain information from context arms regulated as UN (5th), and from individual arm information only (11th and 26th columns).

The same observation that the context information differs between the two datasets applies to the other two rules too (TS and SS). This may be the reason why the dynamic all-arm models perform worse than dynamic one-arm models in all experiments, since

Table 6.4.: Combinations of traffic regulators at intersections in the three datasets. The numbers refer to number of intersections.

Dataset	three-way junctions							four-way junctions						
	UN SS	YS PS	PS SS	UN (all)	SS (all)	TS (all)	Total	UN SS	YS PS	PS SS	UN (all)	SS (all)	TS (all)	Total
Champ.	293	0	0	33	15	9	350	220	0	0	9	52	80	361
Chicago	36	0	0	4	8	8	56	10	0	0	0	17	71	98
Hanover	0	386	5	230	0	88	709	0	82	9	94	0	153	338

Table 6.5.: Combinations of traffic regulators at the intersections of the three datasets, where intersection arms are crossed by at least five straight trajectories. The numbers refer to number of intersections.

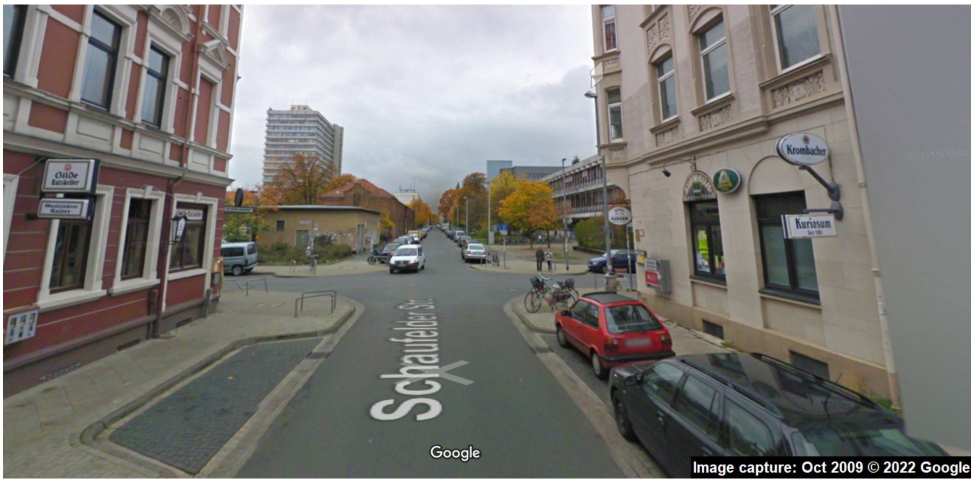
	Three-way intersections															Four-way intersections														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
	TS TS TS	YS YS PS	UN UN UN	UN UN SS	UN UN	YS PS	UN SS	PS PS	SS SS	TS TS	UN	PS	SS	TS	Total	TS TS TS TS	SS SS SS SS	UN UN UN UN	TS TS TS	SS SS SS	UN UN UN	UN UN SS	UN UN	SS SS	TS TS	UN	PS	SS	TS	Total
Champ.	0	0	0	2	115	0	3	0	3	7	38	0	2	0	170	12	1	0	10	0	0	2	70	16	34	30	0	12	10	197
Chic.	0	0	0	0	9	0	0	0	2	3	25	0	6	3	48	0	0	0	2	1	0	0	0	3	20	9	0	11	24	70
Han.	2	1	1	0	30	1	0	101	0	22	14	51	0	5	228	2	0	1	9	0	1	0	8	1	41	19	10	0	16	108

training in learning transferability experiments is done with information that differs compared to that used by the same classifier for predicting. For the Exp3, Exp4, Exp5 and Exp6 the information between the training and predicting datasets differs even more. A classifier trained on the Hanover dataset will use examples for recognising UN regulated arms, which in the dynamic all-arm settings will contain information from context arms that are regulated as UN UN (see 3rd and 21st column in Table 6.3), UN (5th and 23rd columns), UN UN UN (18th column), and individual arm information (11th and 26th columns). For the same training task for UN recognition in the Champaign dataset, the classifier will use examples which contain information from context arms regulated as UN SS (4th and 20nd column), UN (5th and 23rd columns), SS (7th column) and from individual arm UN related information only (11th and 26th columns).

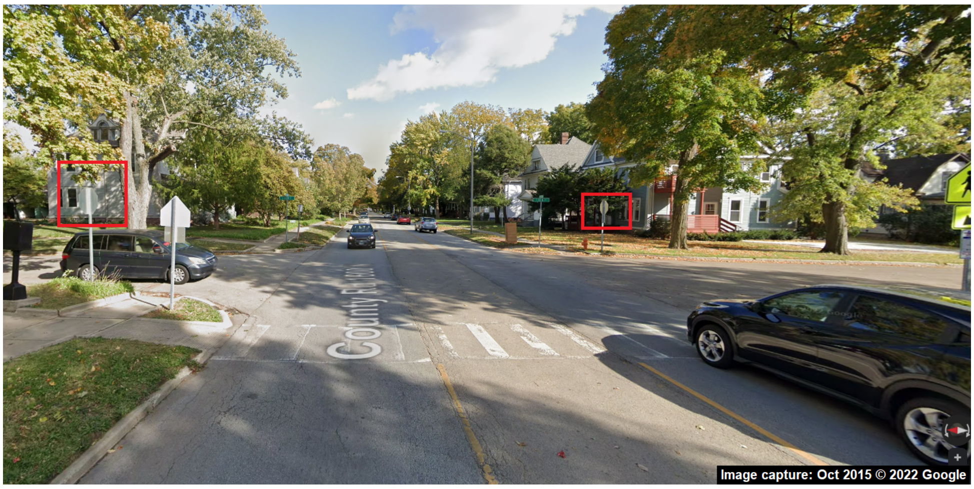
3. (Explanation of observation 4) The fact that the accuracy in Exp3 for the dynamic and hybrid models is low (e.g. 0.35 and 0.41 for the one-arm models respectively), where training is done in Hanover and predicting in Champaign, but in Exp4 where training is done in Champaign and predicting in Hanover, is high (0.92 for both models) can be explained by considering not only on regulation terms but on the movement behaviour imposed by the rules of an intersection.

   More specifically, in Table 6.4, we can see per intersection type (three-way and four-way intersections), the combinations of regulations in the three datasets. Although the Champaign and Hanover datasets have common regulations (UN, TS), the movement behaviour imposed by the rule UN differs in the two cities. In Hanover the UN coexists at the intersection with other UN rules and requires the vehicle to yield for traffic coming from the right, which means that it can decelerate or/and stop to check for traffic coming from the right. Figure 6.12a shows a typical example of a UN controlled intersections in Hanover, where one can see that due of the buildings and the parked vehicles near the intersection, the driver must at least slow down to check for traffic coming from the right. In Champaign and Chicago the UN rule coexists at intersections with SS (UN, UN, SS in three-way intersections and UN, UN, SS, SS in four-way) or with UN (all-way UN for three-way and four-way intersections). The behaviour of vehicles at all-way UN intersections will be similar to that in Hanover. However, at UN-SS controlled intersections, vehicles crossing uncontrolled arms can pass through the intersections unhindered -without slowing down- basically because early before the intersection drivers can check whether the characteristic octagonal stop sign is present on the right intersection arm. By seeing the SS early, drivers don't have to decelerate at all before the intersection. In addition, the physiognomy of Champaign is that, that allows good visibility near intersections, so stop signs as shown in Figure 6.12b, or the stop line as shown in Figure 6.12c are easily seen by drivers.

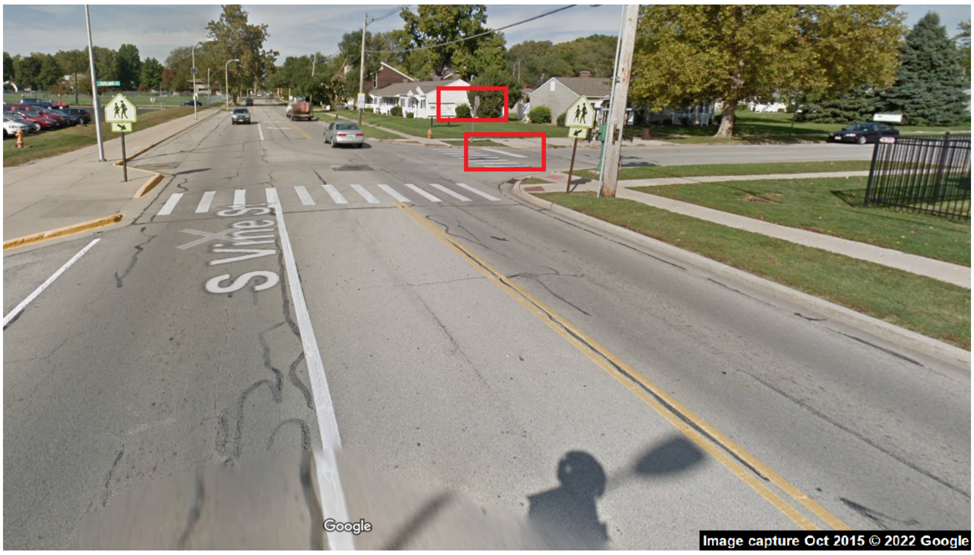
   Therefore, when a classifier is trained on the Hanover dataset, it learns as UN the movement behavior of decelerating/stopping for right-traffic yielding. When this classifier is applied to the Champaign or Chicago dataset (for predicting labels), UN arms that coexist with SS and drivers cross them unhindered, are misclassified as TS instead of UN (Figure 6.13a and 6.13c). However, when the classifier is trained in Champaign or Chicago, it learns to identify as UN the two different types of movement behaviour at UN controlled arms (decelerating/stopping or unhardened). When this classifier is applied to the Hanover dataset, it can recognize UN arms because it has been trained with relevant data (Figure 6.13b and 6.13d). The same observation and explanation holds for Exp5 and Exp6.



(a) Hanover.

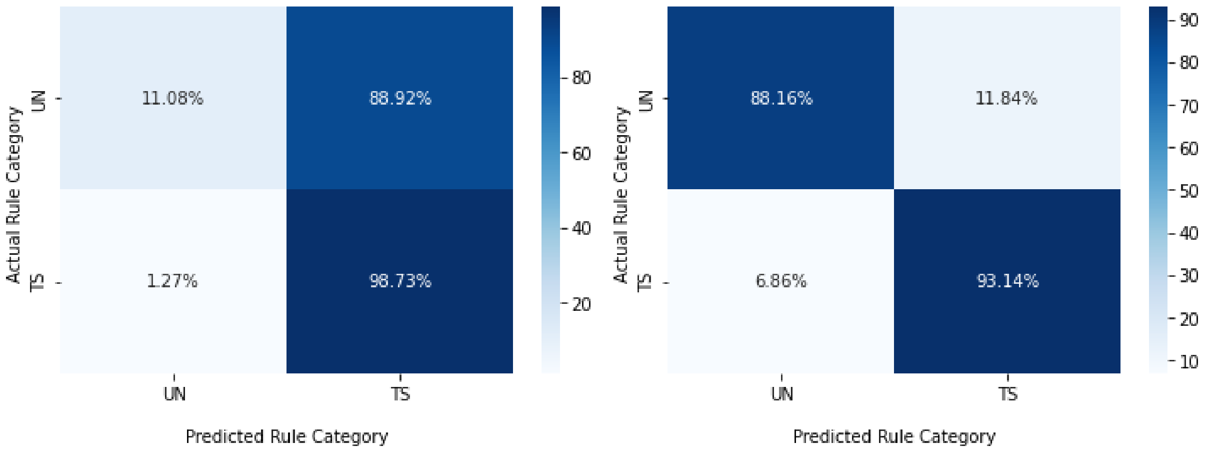


(b) Champaign

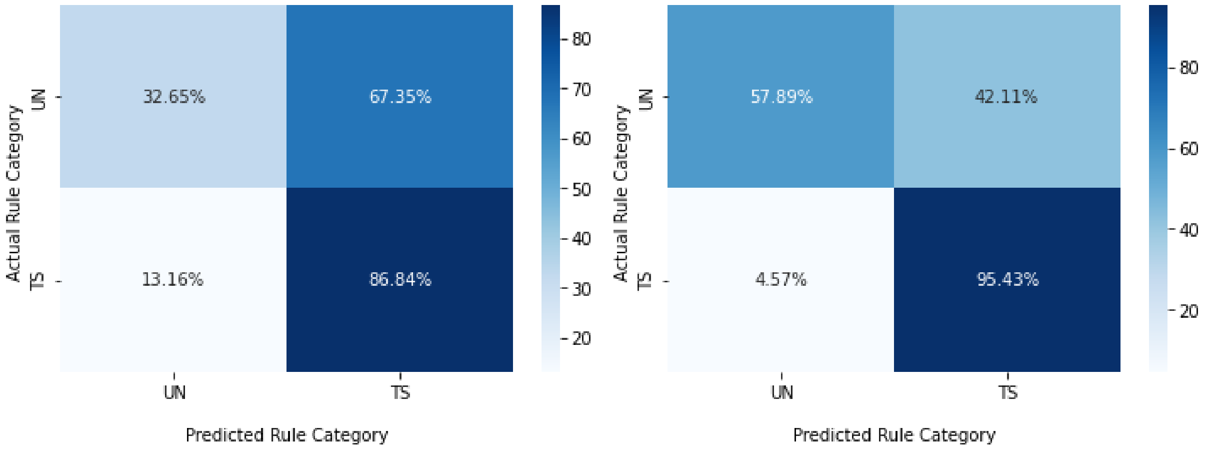


(c) Champaign

Figure 6.12.: Examples of uncontrolled intersections in Hanover (all-way UN) and in Champaign (UN-SS). In red are framed stop signs and a stop line.



(a) Exp3: Train in Han., predict in Champ. (acc: 0.35). *(b)* Exp4: Train in Champ., predict in Han. (acc: 0.92).



(c) Exp5: Train in Han., predict in Chic. (acc: 0.66). *(d)* Exp6: Train in Chic., predict in Han. (acc: 0.84).

Figure 6.13.: Confusion matrices of the dynamic one-arm model for Exp3, Exp4, Exp5, and Exp6.

Table 6.6.: Classification results of the six experiments on the transferability of learning between cities (hybrid all-static approach). Ac: accuracy on the Predict dataset when training is done with data from the city given in the Training dataset column; $\Delta_{Ac} = Ac_{Orig} - Ac$, *where* Ac_{Orig} *is the accuracy when training is done in the same city as the Predict dataset. The results under the "original" columns refer to the experiments which use the GPS data with their original sampling rate. The "undersampled" results, refer to the experiments where the Champaign and Hanover datasets are (under)sampled (see page 131, observation 5, for the settings of undersampling). The best accuracy per experiment is indicated in bold.*

				Hybrid (all-static)					
				original data			undersampled data		
Exp.	Train	Predict	Reg.	Ac	Ac_O	Δ_{Ac}	Ac	Ac_O	Δ_{Ac}
1	Champaign	Chicago	UN, SS, TS	**0.80**	0.91	0.11	0.73	0.91	0.18
2	Chicago	Champaign	UN, SS, TS	**0.78**	0.94	0.16	0.75	0.93	0.18
3	Hanover	Champaign	UN, TS	**0.53**	0.96	0.43	0.46	0.96	0.50
4	Champaign	Hanover	UN, TS	**0.91**	0.98	0.07	0.80	0.98	0.18
5	Hanover	Chicago	UN, TS	**0.82**	0.94	0.12	0.61	0.94	0.33
6	Chicago	Hanover	UN, TS	**0.86**	0.98	0.12	0.85	0.98	0.13

4. (Explanation of observation 5) Table 6.6 shows the classification accuracy of the six experiments, both using the original data and undersampled data. In all experiments, the classification accuracy is greater when the original data is used, which means that undersampling a dataset, so that the two datasets involved in the learning/prediction process have similar sampling rate, does not positively affect the classification accuracy. Therefore, these experiments recommend no subsampling preprocessing of the datasets in the context of learning transferability.

A general conclusion of the above findings and discussions is that cross-city transferability learning is feasible, but with reduced accuracy. For cities of the same country, such as Chicago-Champaign, we found a decrease in accuracy between 11% and 16% (cf. Δ_{Ac} of hybrid all-static in Table 6.3, Exp1 and Exp2). Between cities of different countries, on best case the accuracy is 7% smaller than when the classifier is trained and predicts with data of the same city, and on worst case 43% (cf. Δ_{Ac} of hybrid all-static in Table 6.3, Exp3-Exp6). The hybrid all-static model performed better than the other TRR models in the majority of experiments, and would therefore be recommended as the default model for transferability learning purposes. Moreover, in terms of transferability of learning between cities of different countries, the four experiments Exp3, Exp4, Exp5 and Exp6 showed that learning has some limitations stemming from the fact that they were conducted between a subset of regulators (UN, TS) that Hanover and Champaign and Hanover and Chicago have in common, rather than across the entire set of regulators as was done in Exp1 and Exp2. In a real-world scenario, eliminating the traffic regulation task in two regulation categories would be neither useful nor realistic. However, even in a scenario where the regulators of the two cities are only partly the same and the same regulator may impose different traffic behaviors (such as UN in Hanover and UN in Champaign and Chicago), learning transferability can be feasible under the certain training conditions discussed previously (see discussion on Exp3 and Exp4). In addition, the fact that the reason that learning was not successful in Exp3 and Exp5 was the different context of intersection rules between the cities, (all-way UN controlled intersections in Hanover and all-way UN and UN-SS controlled intersections in Champaign/Chicago), can support the argument that transferability of learning between cities in different countries can be possible when cities share the same intersection rules (in contrast to the scenario with partly the same regulations tested in the experiments between Hanover and Champaign and between Hanover and Chicago).

6.5. Incremental (Online) Learning

Here we propose a hypothetical TRR incremental learning scenario, depicted in Figure 6.14. To the author's knowledge, such a scenario has not been explored to date. Suppose a mobile phone application which, upon detecting driving motion, starts recording GPS traces. The traces are uploaded to the application's server where features are extracted and data instances are generated and request real-time predictions for traffic regulations. The mobile application also has a *participatory* component, through which drivers can volunteer to answer simple voice questions such as "are you now crossing an intersection with a traffic light?", with a "Yes" or "No". The application records the response and encodes it as a real-time label of the referenced intersection arm, which is used directly for evaluating the current TRR learning component and incrementally training it as well.

In contrast to batch learning, which has been used in all the studies presented so far in this thesis, incremental learning deals with data streams. As Read et al. (2012) point out, a data stream has also different requirements from the traditional batch learning setting,

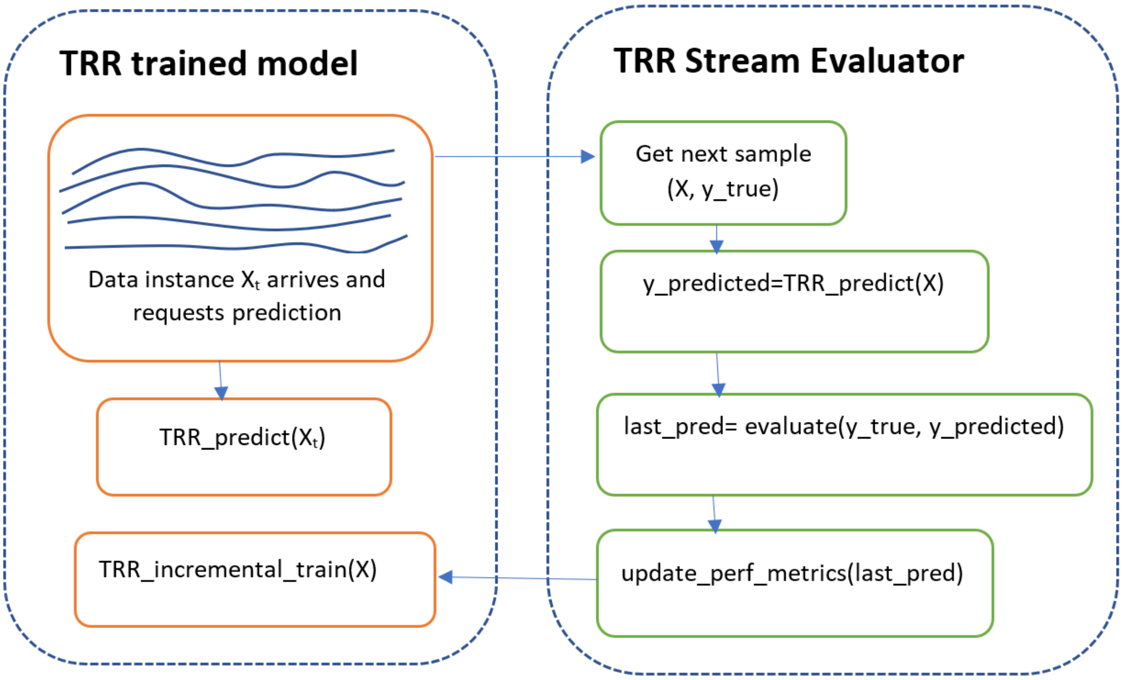
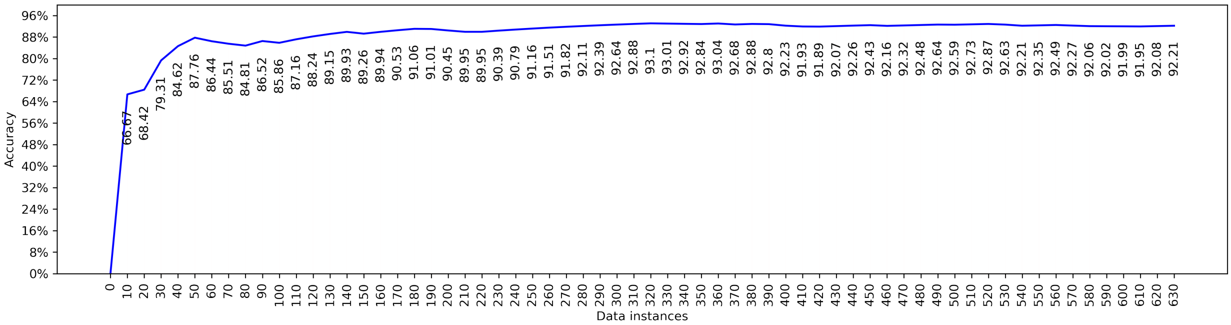

TRR trained model
Data instance X_t arrives and requests prediction
TRR_predict(X_t)
TRR_incremental_train(X)
TRR Stream Evaluator
Get next sample (X, y_true)
y_predicted=TRR_predict(X)
last_pred= evaluate(y_true, y_predicted)
update_perf_metrics(last_pred)

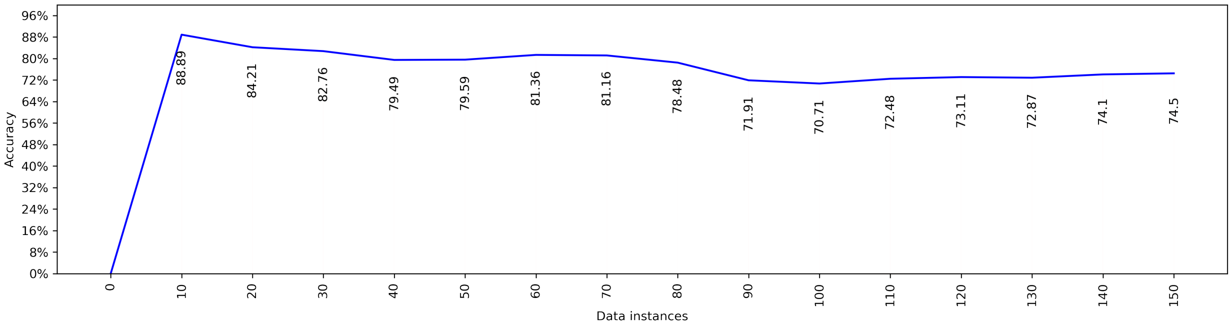
Figure 6.14.: A TRR incremental learning scenario.

with most significant be the following: (1) be ready to predict at any point; (2) data may be evolving over time; and (3) expect an infinite stream, but process it under infinite resources (time and memory). An instance of a data stream in the context of TRR is a description of an intersection arm, encoded with features extracted from the trajectories and OSM, that can arrive at any time to request a prediction. The classifier then provides a prediction for the traffic regulation of the requested data instance (arm) using its current learning model (Figure 6.14, left component). Labeled data can also arrive at any time. A prediction is then made using the current trained model. The current learning model is then evaluated by comparing its latest prediction with the actual label of the data instance. The classifier performance is updated to reflect the current classification performance and as the last step of this online process, the learning model is trained with the data instance (Figure 6.14, TRR_incremental_train(X)). For the implementation of this scenario, the Python machine learning library *River* (Montiel et al., 2020) and in particular the function *ensemble.AdaptiveRandomForestClassifier* was used.

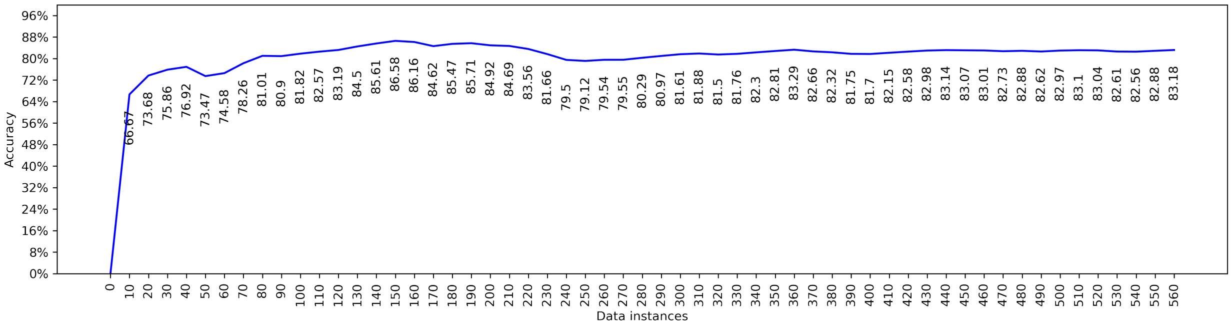
Figure 6.15 shows the classification accuracy of the incremental TRR model for the three datasets. A prediction is made *every* time a data instance arrives. However in Figure 6.15, the accuracy is plotted every 10 data instances in order to have all the predictions of the data stream of each dataset on one graph. The performance for each data instance is given in the Appendix on page 155 (Figure A.2, A.3, A.4, A.5, A.6). A first observation from the three graphs is that compared to the performance of the corresponding default batch learning model (Chapter 5), the incremental classifier performs worst, which is a known advantage of batch learning over incremental learning (Carbonara and Borrowman, 1998). More specifically, on the Champaign dataset, the classifier after seeing 630 examples achieves an accuracy of 92%, while the default model in the same dataset achieves an accuracy of 95% (3% difference, cf. Table 5.2). In the Chicago dataset the difference is more pronounced. After seeing 150 examples, the online model has an accuracy of 75%, while the corresponding batch model has an accuracy of 91% (16% difference). In Hanover, after seeing 560 examples, the incremental model has an accuracy of 83%, while the batch model has 95% (12%



(a) Champaign.



(b) Chicago.

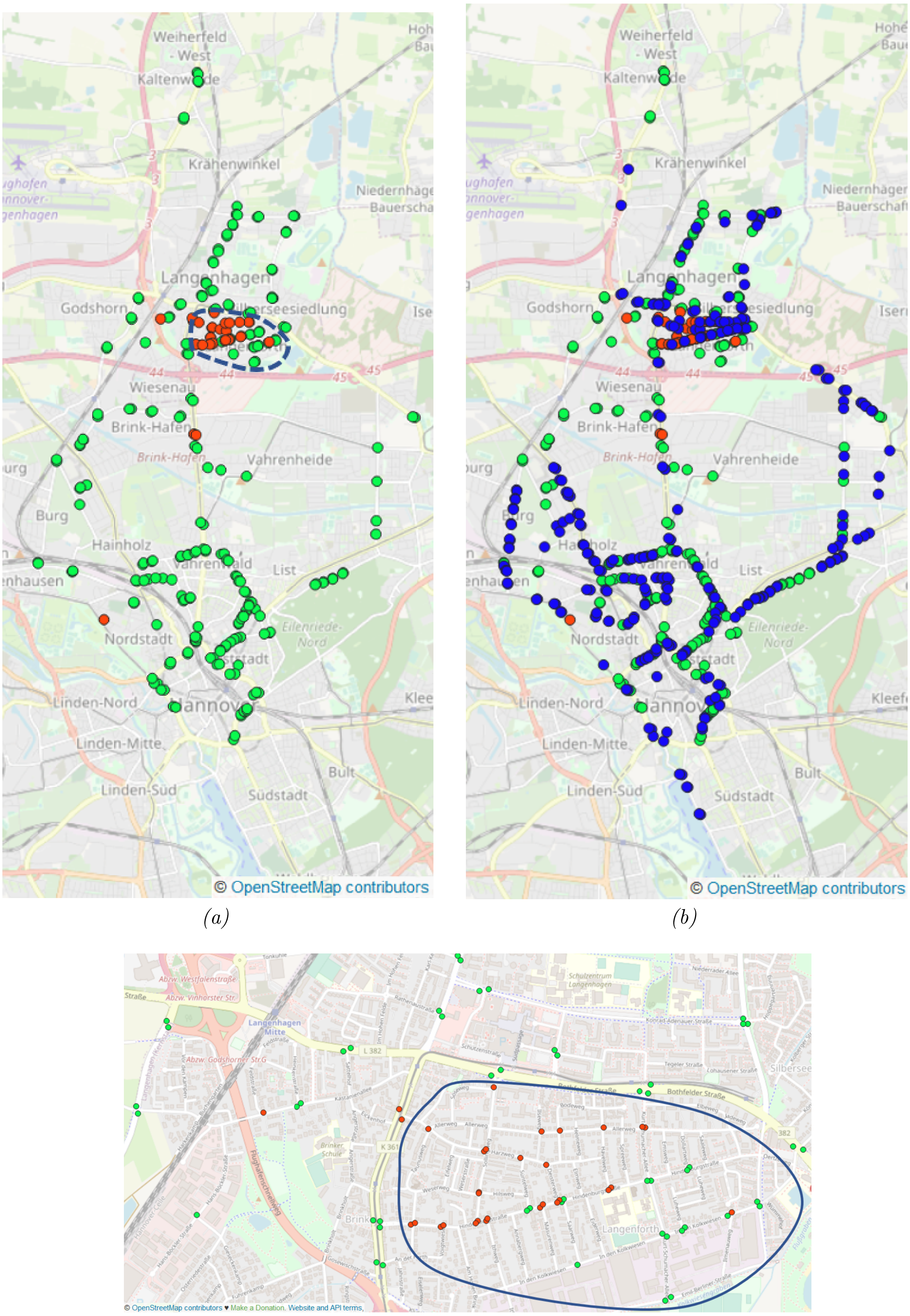


(c) Hanover.

Figure 6.15.: Classification performance in the three datasets under a TRR incremental learning scenario.

difference). Also, in Champaign and Hanover convergence is observed after 250 examples. Chicago has fewer than 250 examples, so no such conclusion can be drawn.

Furthermore, in Hanover around the 220th example the accuracy starts to drop until the 250th example where it starts to increase again slowly (more precisely the accuracy falls between 221-254, Figure A.3). Visualising the examples from 1 to 220 (green spheres in Figure 6.16a) and from 221-254 (red spheres, Figure 6.16a and Figure 6.16c), we can see that the accuracy drops because consecutive predictions are made within a residential area with few examples having been used for training from this area until that time point (18 green examples versus 30 red examples). Also the intersections in this area are mostly uncontrolled intersections where drivers drive at very low speeds as one intersection from the other is very close (cf. with the intersections outside the circled area of Figure 6.16c). We understand the drop in performance as an inefficacy of the classifier to predict regulations



(a) (b)

(c)

Figure 6.16.: Data instances of the Hanover dataset in order of arrival (1 is the first instance of the stream to arrive, 2 is the second, etc.): from 1 to 220 data instances (green spheres), 221-254 (red spheres) and 255-566 (blue spheres). The circled area shown in (c) corresponds to the dotted circled areal indicated in (a).

of such street context due to incomplete training. After the 254th example, the accuracy starts to increase and since many predictions from the same area follow (see blue examples in Figure 6.16b), we can conclude that the classifier learns better after trained with the examples indicated in red.

The goal of implementing these experiments was only to show how the TRR problem could be handled in a data streaming environment. The data streams used in the above experiments are hypothetical and their properties do not reflect the properties that a real TRR data stream would have. A real TRR data stream would be an infinite stream of millions of data instances per day, covering a large number of intersections and evolving over time, allowing for dynamic detection of changes in regulations.

In addition, under the concept of *concept drift* discussed in page 61, intersections that have time-dependent settings (e.g., traffic lights that during the night turn into uncontrolled intersections) could also be detected. Clearly, due to the limitations of the datasets, no further conclusions can be drawn on issues related to change detection, suitability of batch size[3], etc. However, the idea of dynamic TRR, as proposed in this paragraph, could also be explored in more *fluidly* regulated environments, such as shared spaces. Public spaces are designed to create spaces that are primarily pedestrian-friendly and incorporate relevant design elements to pursue this goal. As a result, streets that incorporate the principles of shared spaces tend to have reduced vehicle speeds, increased pedestrian safety and accessibility, and environmental quality (Beitel et al., 2018). The most successful shared space systems are based on simplicity, which is achieved by removing conventional barriers, bollards and signs (Beitel et al., 2018). Although there are no formal traffic regulations, such as stop signs and traffic signals, they impose restrictions on traffic, such as the low speed limit mentioned earlier. Therefore, an interesting research direction would be to detect *hidden* traffic regulations in an incremental manner on streets that incorporate shared spaces.

6.6. Summary

In this chapter, the default TRR model was tested under different scenarios of availability of labeled data. The scenario of no availability of labeled data was explored under the concept of clustering and transferability of learning between cities. It was found that clustering cannot provide accurate predictions and that learning can be transferable between cities, but with reduced accuracy than if the classifier predicted regulators from the same city that it had been trained on. The scenario of availability of a small amount of labeled data was explored under the umbrella of self-training, active learning and cluster-then-label (with/without label-propagation). The most accurate predictions of the above tested learning methods were achieved through active learning, which was found to reduce the number of required labeled data for training by 66.7% on the two datasets used for testing. Finally, a hypothetical scenario was described where trajectory data arrives as data streams and predictions are made in an on-line manner. A possible extension of the idea of incremental TRR in the context of shared spaces was also discussed.

---

[3] Here we tested *instance-incremental* learning (Read et al., 2012), where training is done with only one example. Another option would be *batch-incremental* learning, where training is done in batches of data.

7. Conclusions and Outlook

7.1. Research Questions Addressed in this Thesis

This thesis addressed the traffic regulation recognition (TRR) problem, under different trajectory and classification feature settings (number of trajectories, one-arm features vs. all-arm features), which to author's knowledge has not been done before. Three datasets were used for testing the proposed methodology from the cities of Champaign (US), Chicago (US) and Hanover (DE). The datasets concerned three-way and four-way intersections controlled by the following traffic regulations: traffic signals (TS), stop signs (SS), priority signs (PS) and uncontrolled intersections (UN). More specifically:

1. It presented a new methodology for TRR by analysing GPS traces, where in the classification feature vector, information from context intersection arms (i.e., arms belonging to the same intersection) is also included (all-arm models).
    - It proposed a modification of a well-known clustering technique for detecting short-term movement events such as stopping and slowing episodes.
    - It provided an extensive evaluation of the proposed methodology, which was missing from the literature of TRR from GPS traces. Three datasets that include different regulation types collected from different cities were used for testing the proposed methodology. It was found that including information in the feature vector from context intersection arms proved beneficial for classification: all-arm models outperformed single-arm models (one-arm models use information only from one intersection arm). Moreover, the hybrid all-static model, which combines data from trajectories and map data (here OSM), found to provide accurate results for regulation sets consisting of controlled intersections of UN, SS, PS, and TS: 97% accuracy in Champaign and Hanover, and 95% in the smaller Chicago dataset. Compared to state-of-the art methodologies, such that of Hu et al. (2015), the proposed methodology provided more accurate predictions (acc: 91% vs. 97%, measured in the same dataset-Champaign). Also compared to the latest published work on the field (Cheng et al., 2022) (see page 74), which uses the same Hanover trajectory dataset used in this thesis, as well as satellite images, fed in a Conditional Variational Autoencoder, this thesis methodology provided better classification performance (acc: 90% vs. 97%, measured in the same dataset-Hanover).
    - It proposed an additional consistency check of the predicted labels at an intersection level, correcting misclassified regulators when possible. It was found that domain knowledge rules in general can help recover incorrect predictions in cases where there are predictions available from the context arms and there is inconsistency in the predicted labels, expressed (and detected) as a significant difference in the predicted probabilities of the labels (threshold value for probability difference used: 0.14). If all regulations of the same intersection are predicted with high probability, then unless there is a majority regulation label agreement, no recovering action can be triggered. By applying domain knowledge rules, there

is a gain in accuracy between 1% and 3%, but more importantly, accurate predictions can be made on a number of arms with incomplete data corresponding to 29%-49% of the original data. Interestingly, the (incomplete) arms predicted solely based on the information of context arms are predicted with high accuracy: 99% in Champaign, 98% in Chicago and 100% in Hanover. In addition, the FPR of the class with the highest FPR decreases between 15.6% to 60% (15.6% in Chicago, 60% in Champaign and 58.3% in Hanover), validating the proposal of this thesis to use domain knowledge rules to both recover misclassified arms and to predict regulators for arms without trajectory data. After applying such rules the classification accuracy in the three dataset was increased to: 97% (from 96%) in Champaign and Hanover and 95% (from 92%) in Chicago.

   - It explored the effect of sampling rate on the classification performance. It was found that a low sampling rate 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 1-2% when the sampling interval was doubled from 2 s to 4 s.

2. It investigated the classification performance of the proposed TRR method under different trajectory settings.

   - It explored the effect of turning trajectories on classification performance. It was found that the negative effect on accuracy, as validated by the two larger datasets, when both straight and curved trajectories are used, is between 1%-3%. Therefore, the exclusion of curved trajectories has a positive effect on classification performance.

   - It examined the minimum number of trajectories per intersection required to achieve optimal accuracy. It was found that the optimal number of trajectories per intersection arm for the computation of classification features is five straight trajectories. Moreover, restricting the number of tracks to a certain number negatively affects the classification performance. With only three straight trajectories per intersection arm, classification accuracy is equal to or greater than 85% across all datasets (85% in Chicago, 89% in Hanover and 92% in Champaign). With five straight trajectories, the accuracy is equal to or greater than 90% (90% in Chicago and 92% in Champaign and Hanover).

3. It investigated the classification performance of the methodology under sparsely labeled data. More specifically, it explored the following possible solutions: clustering, self-learning, active-learning, cluster-then-label, and learning transferabilty between cities. It was found that clustering cannot provide accurate predictions and that learning can be transferable between cities, but with reduced accuracy than if the classifier predicted regulators from the same city that used for training. The most accurate predictions of the above tested learning methods were achieved through active learning. It was found to reduce the number of required labeled data for training by 66.7% on both datasets used for testing.

4. Finally, a hypothetical scenario was described where information arrives as data streams. Instead of processing all data at one time for building the TRR learning model, one observation is processed at a time and the learning model is updated incrementally. A possible extension of the idea of incremental TRR in the context of shared spaces was also discussed.

7.2. Outlook

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 behaviour 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 under various settings of missing OSM data. In the three datasets tested in this dissertation, very often speed-related data were missing from OSM, but the performance remained high. It would be interesting to investigate under what conditions of missing data, the performance would be affected to the extent that, for example, the dynamic model would be preferable to the hybrid model. Furthermore, an alternative way to those proposed in this thesis for addressing the problem of sparsely labeled data, would be to use transfer learning, where few labeled data from the target city would be used to determine the local context, given that the classifier would have acquired its learning capability from (non sparsely) data obtained from other than target city.

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 done 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 study 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, intersection features could be identified that make certain locations (and perhaps classes of regulations) easier to identify, regarding their traffic control, than other locations (or regulation categories). Also, the results of this analysis, could support possible methodological interventions on the thesis proposed methodology.

Furthermore, the way in which the trajectory density 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. a section of a city is sampled from dozens of trajectories and other parts from only a few tracks. This aspect is not taken into account in the current settings of splitting the datasets into training and test sets. Perhaps splitting the dataset taking this aspect into account would provide even better results. Moreover, the accurate predictions of traffic regulations that the proposed methodology showed to be capable of, can motivate the idea to crowd-source additional rule categories that can be added to maps and exploited by location-aware applications. The latter in 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, Krisp and Keler (2015) 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 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.

List of Acronyms

ADAS Advance Driver Assistance Systems

CB-SDoT Clustering Based Detection of Stop and Deceleration Episodes of Trajectories

CB-SMoT Clustering Based Stops and Moves of Trajectories

CC Conventional Crowdsourcing

cf. compare

CNN Convolutional Neural Network

CVAE Conditional Variational Autoencoder

DB Database

FNR False Negative Rate

FPR False Positive Rate

GB Gradient Boost

GIS Geographic Information System

GNSS Global Navigation Satellite Systems

GPS Global Positioning System

i.e. that is

LBS Location Based Services

NN Neural Network

OSM OpenStreetMap

PS Priority Sign

RF Random Forest

RoI Region of Interest

SSL Semi-Supervised Learning

SC Spatial Crowdsourcing

SS Stop Sign

TNR True Negative Rate

TPR True Positive Rate

TRR Traffic Regulation Recognition

TS Traffic Signals

UN Uncontrolled

YS Yield Sign

Index

A
accidents, 17
accuracy, 51
active learning, 125
ADAS, 4
air pollution, 4
all-arm model, 97
animal behaviour, 29
anomaly detection, 2
automatic map update, 3
automatic road network extraction, 3
autonomous driving, 3
autonomous vehicles, 4

B
bagging, 48
batch learning, 61, 137
BeiDou, 24
bicycle paths' roughness, 2
binary classifier, 42
BIRCH, 30
boosting, 48
bootstrap, 48
border point, 55
braking incidents, 2
bumps, 2

C
cascading patterns, 30
CB-SMoT, 34
change detection, 3
classification, 41, 42
cluster, 29, 34
clustering, 30, 31, 34, 41, 52, 121
cluster-then-label, 127
collective behaviour, 27
common sequences, 34
complete clustering, 53
complexity rating, 6
complicated crossings, 6
computer vision, 10
conceptual view (trajectories), 2
conflict points, 19
confusion matrix, 50
context arms, 97
contextual data, 33
core point, 55
coupling patterns, 30
crossroad, 17
crowd-sensing, 1
crowd-sourcing, 10

D
data anomalies, 31
data cleaning, 31, 32
data fusion, 4
data integration, 31
data reduction, 31
data stream, 61, 137
data transformation, 32
DBSCAN, 31, 34, 54
decision tree, 31
deduction, 42
density-based clustering, 54
deterministic classifier, 42
detour, 26
discriminative classifier, 43
domain expert, 31
downsampling, undersampling, 25
driver intention, 4
driving analysis, 4
driving behaviour, 12
DTW, 30

E
earthquake warning, 2
eco-friendly routing, 4
ecology, 29
ensembles, 46
entity, 27
exclusive clustering, 53
exploratory analysis, 31

F
F1-score, 52
fatalities, 17
feature engineering, 41
flock, 29
FNR, 51
FPR, 51
framework, 31, 32
fuel consumption, 4
fuel-efficient maps, 4
fuzzy clustering, 53
Fuzzy C-Mean, 31

G
Galileo, 24
generative models, 43
geodata, 1
geographic data, 1
Geolife(dataset), 25
geospatial data, 1
gini score, 45
global classifier, 43
GLONASS, 24
GNSS, 24
google maps, 4
google street view, 7
GPS log, 2
GPS navigator, 26
GPS receiver, 10
GPS traces, tracks, 3, 10
GPS, systematic errors, bias, random errors, 23
GPX, 25
green maps, 4

H
Hausdorff distance, 30
hidden patterns, 2
hierarchical (nested) clustering, 53
historic traffic patterns, 4
honking, 2
human activities, 2
hypothesis, 12

I
impurity score, 45
incremental (online) learning, 61, 137
induction, 42
interchange, 17
interesting locations, places, 2, 29

intersection, 17
intersection arm, approach, 68
intersection design, 19
intersection detection, 3
intersection travel time, 2
iPhone, 10

J
J48, 31

K
kernel density estimation, 31
K-means, 34, 53, 127

L
LCSS, 30
learning algorithm, 42
learning rate, 49
learning types, 41
line simplification, 32
linear classifier, 43
linear neighborhood, 36
local classifier, 43

M
machine learning, 40
macro average score, 52
map update, 3
mapillary, 7
map-matching, 32
meeting point, 29
mobile mapping, 7
modAL (Python library), 126
move, 35
movement attractor, 29
movement pattern, 2, 12, 27, 29
multiclass classifier, 42
multi-way stop, 20

N
Naive Bayes, 31
navigation devices, 3
noise point, 55
nonlinear classifier, 43

O
object recognition, 10
one-arm model, 97
one-way stop, 20
opportunistic sensing, 1
OSM, 10
outliers, 30
overalapping clustering, 53

P
partial clustering, 53
participatory sensing, 1
partitional (unnested) clustering, 53
pasting, 48
path, 26
pattern, 2, 29
pattern recognition, 2
PCA, 30
performance metrics, 49
periodic pattern, 29
periodicity, 29
personalised routing, recommendations, 6
PostGIS, 93
PostgreSQL, 93
precision, 51
pre-processing, 32
priority control, sign, 20
probabilistic classifier, 42
pseudo-labels, 58
Python, 93, 102, 124, 126, 138

R
random forest, 31, 48
random patches, 48
random subspaces, 48
randomised trees, 48
recall, 51
RedMiner, 32
region of interest, 29
regression, 41
regulation-aware maps, 5
reinforcement learning, 42
risk assessment, 4
River library (Python), 138
road names and classes, 2
road network inference, 3
ROI, 29
roundabout, 19, 20, 22
route, 26
route recommendation, 4, 6
routepoint, 26

S
sampling frequency, rate, period, 24
satellites, 22
Scikit-Learn, 45, 49
scissor cross, 19
self-learning, 58
self-training, 58, 124
semantic enrichment, 33
semantic trajectories, 2, 33, 34
semantics, 32, 33
semi-supervised learning, 42, 57
sensitivity, 51
sensors, 1
shared spaces, 141
signal lost, 24
signalized intersections, 22
significant locations, 2
similarity distance measures, 30
skewed cross, 19
sklearn (Python library), 45, 93, 124
smartphone, 1, 10
social media, 2
sparsely labeled data, 121
spatial crowdsourcing, 1
spatiotemporal change, 31
spatiotemporal data, 27, 29
spatiotemporal data mining, 29
spatiotemporal hotspots, 31
spatiotemporal partiotioning, 30
spatiotemporal prediction, 30
spatiotemporal samples, 3
spatiotemporal summarization, 30
spatiotemporal trajectory, 2
specificity, 51
speed-profiles, 71
sport scene analysis, 29
ST-DBSCAN, 30
stop episode, 34
stop sign, 20
stops and moves, 2, 34
street level photos, 7

StVo, 20
sudden events, 2
supervised learning, 41
surveillance, 29
surveying, 6

T
tele-coupling patterns, 30
T-intersection, 19
TNR, 51
TPR, 51
track, 26
trackpoint, 26
traffic condition, 2
traffic control signals (traffic lights), 20, 22
traffic participants, 19
traffic regulations, rules, controls, regulators, 3, 4, 20
traffic sign recognition, 10
traffic signs, 20
trajectory, 2, 25, 27
trajectory behavior, 27
trajectory compression, 32
trajectory mining, 32
trajectory pattern, T-pattern, 27
transferability, 129
travel time, 2
trip arrival prediction, 2
two-way stop, 20

U
uncontrolled intersection, 20
unsupervised learning, 41, 52
up-to-date maps, 3
urban sensing, 2
urban transport, 2

W
waypoint, 26
weighted average score, 52
Weka, 32

X
XGBoost, 49, 93, 102, 127
X-intersection, 19
XML, 25

Y
yield control, sign, 20
Y-intersection, 19

A. Appendix

*Table A.1.: Classification results of the hybrid one-arm model (*GB*).*

Dataset	Label	Recall	Precision	F-Measure	Accuracy	Support
Champaign	UN	0.97	0.96	0.97		424
	SS	0.89	0.87	0.86		52
	TS	0.90	0.94	0.92		157
	W.Avg.	0.95	0.95	0.95		633
					0.95	
Chicago	UN	0.80	0.86	0.81		49
	SS	0.83	0.83	0.82		29
	TS	0.84	0.83	0.83		76
	W.Avg.	0.82	0.84	0.82		154
					0.82	
Hanover	UN	0.84	0.71	0.76		76
	PS	0.89	0.91	0.90		315
	TS	0.89	0.94	0.91		175
	W.Avg.	0.88	0.89	0.89		566
					0.88	

*Table A.2.: Classification results of the hybrid all-static model (tuned *GB*).*

Dataset	Label	Recall	Precision	F-Measure	Accuracy	Support
Champaign	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	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	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	

Table A.3.: Classification results of the reduced and default models.

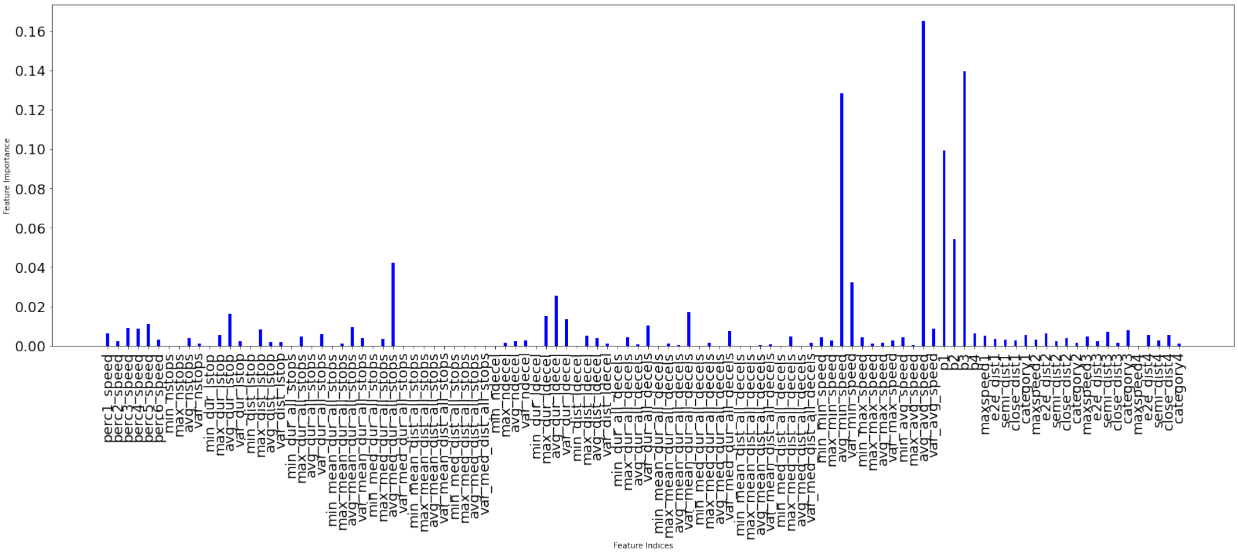
Dataset	Label	Recall	Precision	F-Measure	Accuracy	Support
Champaign (3-way)	UN	0.99	0.98	0.98		267
	SS	0.75	0.80	0.77		12
	TS	0.82	0.90	0.81		25
	W.Avg.	0.96	0.96	0.96		304
					0.96	
Champaign (4-way)	UN	0.99	0.95	0.97		157
	SS	0.88	1.0	0.92		40
	TS	0.95	0.96	0.95		132
	W.Avg.	0.96	0.96	0.96		329
					0.96	
Champaign (default)	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 (3-way)	UN	0.98	0.95	0.95		41
	SS	0.80	0.70	0.73		10
	TS	0.70	0.70	0.70		9
	W.Avg.	0.92	0.88	0.89		60
					0.92	
Chicago (4-way)	UN	0.80	0.60	0.67		8
	SS	0.75	0.90	0.80		19
	TS	0.95	0.96	0.96		67
	W.Avg.	0.93	0.94	0.92		94
					0.93	
Chicago (default)	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 (3-way)	UN	0.96	1.0	0.97		45
	SS	0.98	0.96	0.97		277
	TS	0.83	0.92	0.87		65
	W.Avg.	0.95	0.96	0.95		387
					0.95	
Hanover (4-way)	UN	0.90	1.0	0.94		31
	PS	0.92	0.98	0.94		38
	TS	0.99	0.95	0.97		110
	W.Avg.	0.96	0.97	0.96		179
					0.96	
Hanover (default)	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	

Table A.4.: Classification results of the hybrid all-static model after applying consistency check with domain knowledge rules.

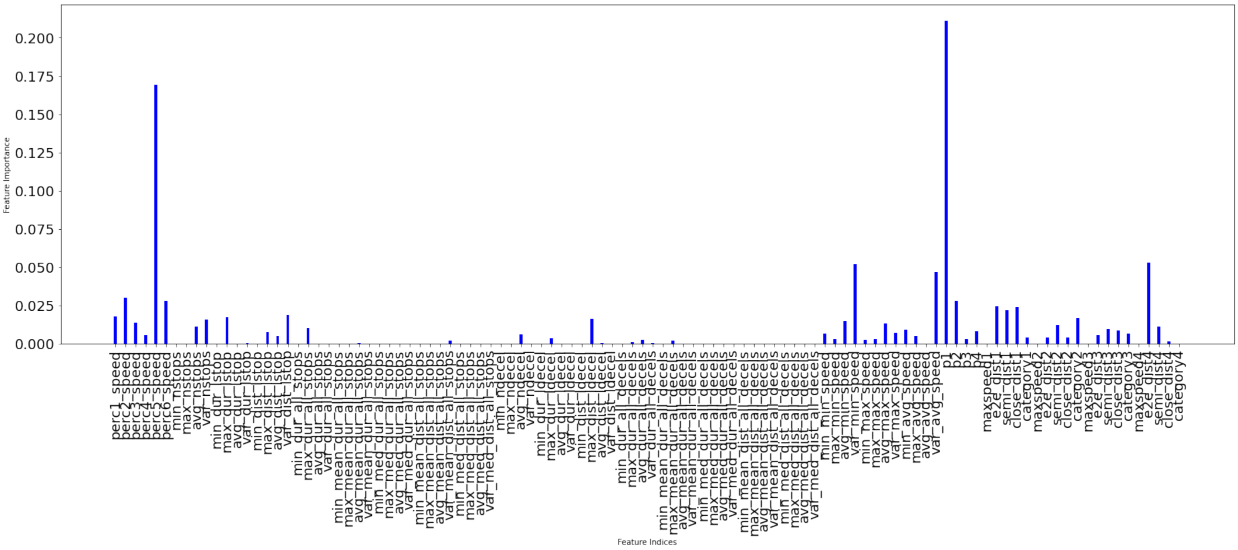
Dataset	Label	Recall	Precision	F-Measure	Accuracy	Support
Champaign	UN	0.96	0.99	0.97		426
	SS	0.97	0.97	0.97		285
	TS	0.98	0.91	0.94		237
	W.Avg.	0.97	0.97	0.97		948
					0.97	
Chicago	UN	0.96	0.94	0.95		49
	SS	0.93	0.88	0.90		43
	TS	0.95	0.97	0.96		109
	W.Avg.	0.94	0.94	0.94		201
					0.95	
Hanover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

Table A.5.: Clustering results (K-means) of the hybrid all-static model.

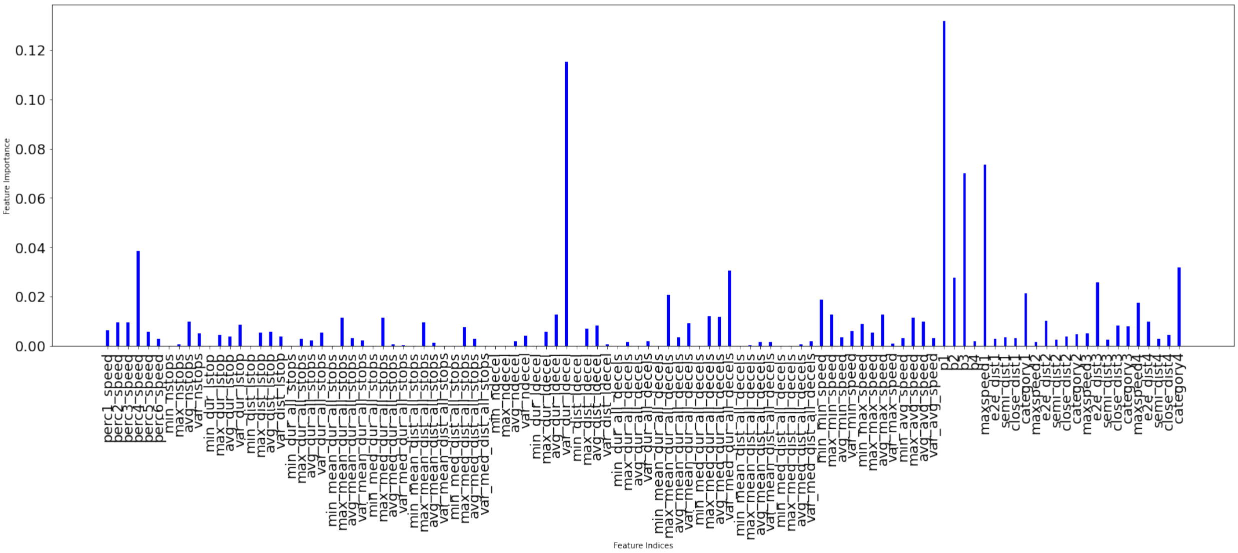
Dataset	Label	Recall	Precision	F-Measure	Accuracy	Support
Champaign	UN	0.89	0.91	0.90		424
	SS	0.00	0.00	0.00		52
	TS	0.62	0.71	0.66		157
	W.Avg.	0.75	0.79	0.77		633
					0.79	
Chicago	UN	0.65	0.73	0.69		49
	SS	0.63	0.83	0.72		29
	TS	0.90	0.72	0.80		76
	W.Avg.	0.77	0.75	0.75		154
					0.75	
Hanover	UN	0.29	0.07	0.11		76
	PS	0.73	0.90	0.81		315
	TS	0.75	0.71	0.73		175
	W.Avg.	0.68	0.73	0.69		566
					0.73	



(a) Champaign.



(b) Chicago.



(c) Hanover.

Figure A.1.: Feature importance.

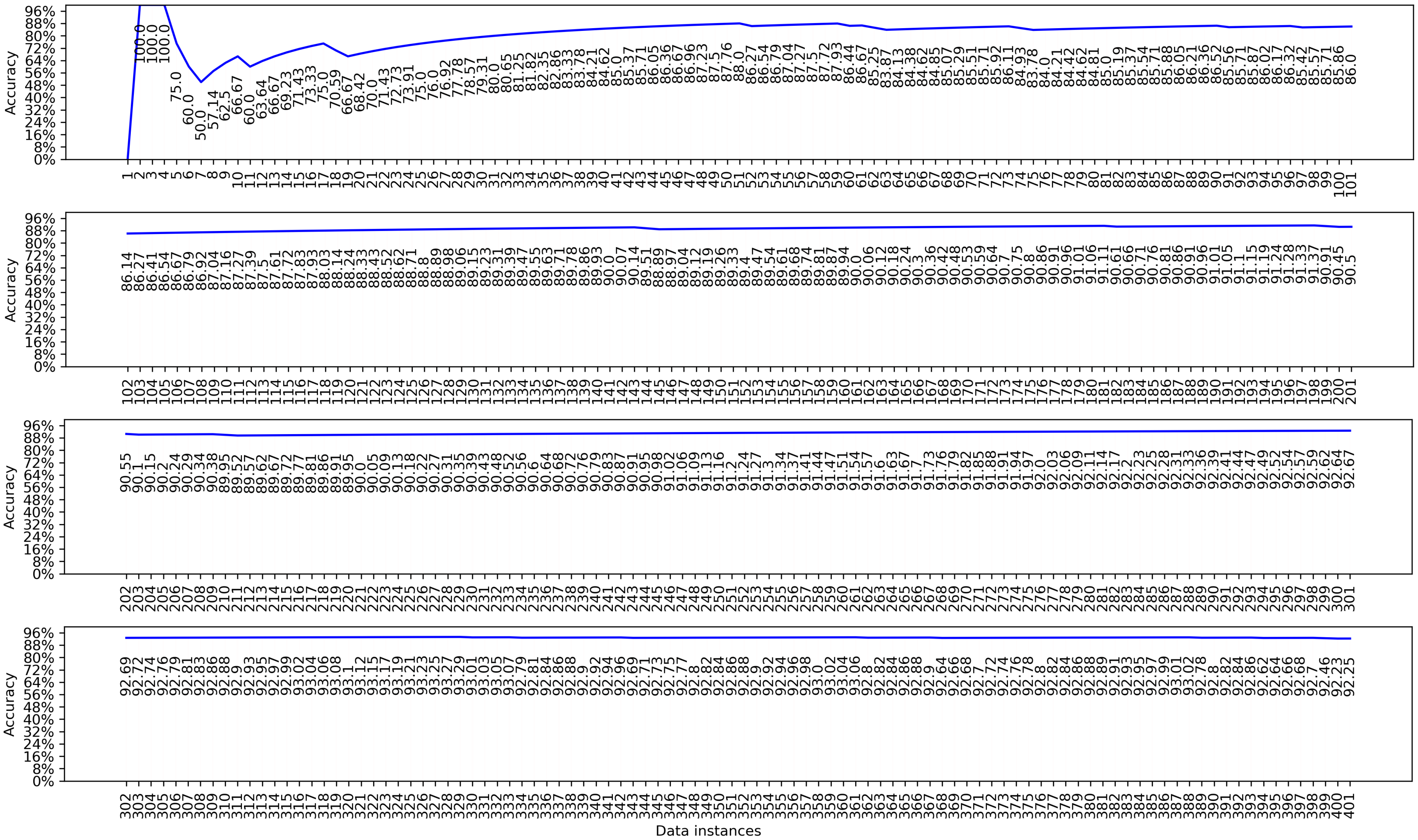


Figure A.2.: Incremental learning in the Champaign dataset (figure continues on the next page).

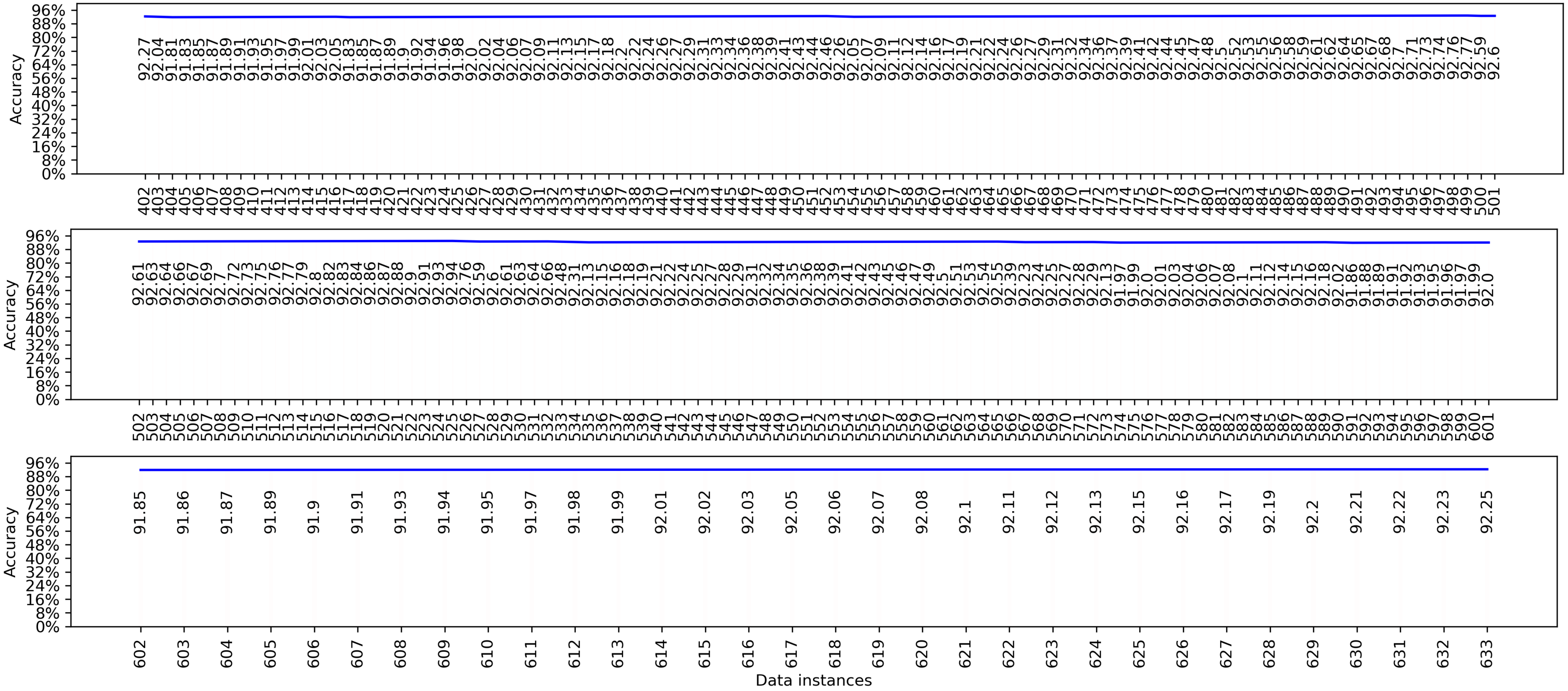


Figure A.3.: Incremental learning in the Champaign dataset (continuing from the previous figure).

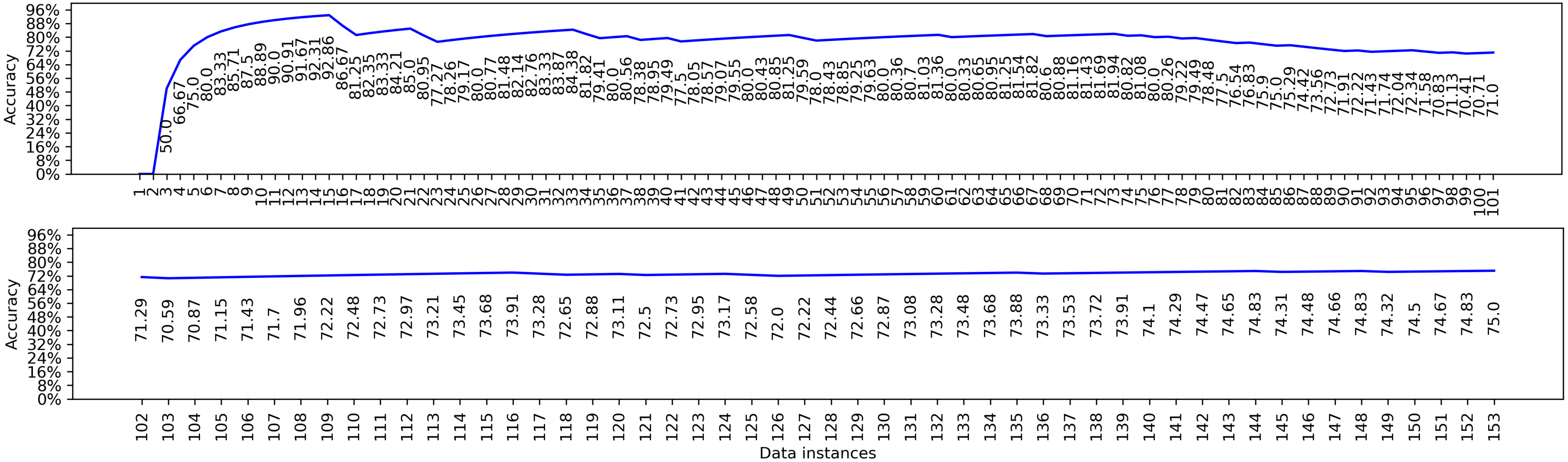


Figure A.4.: Incremental learning in the Chicago dataset.

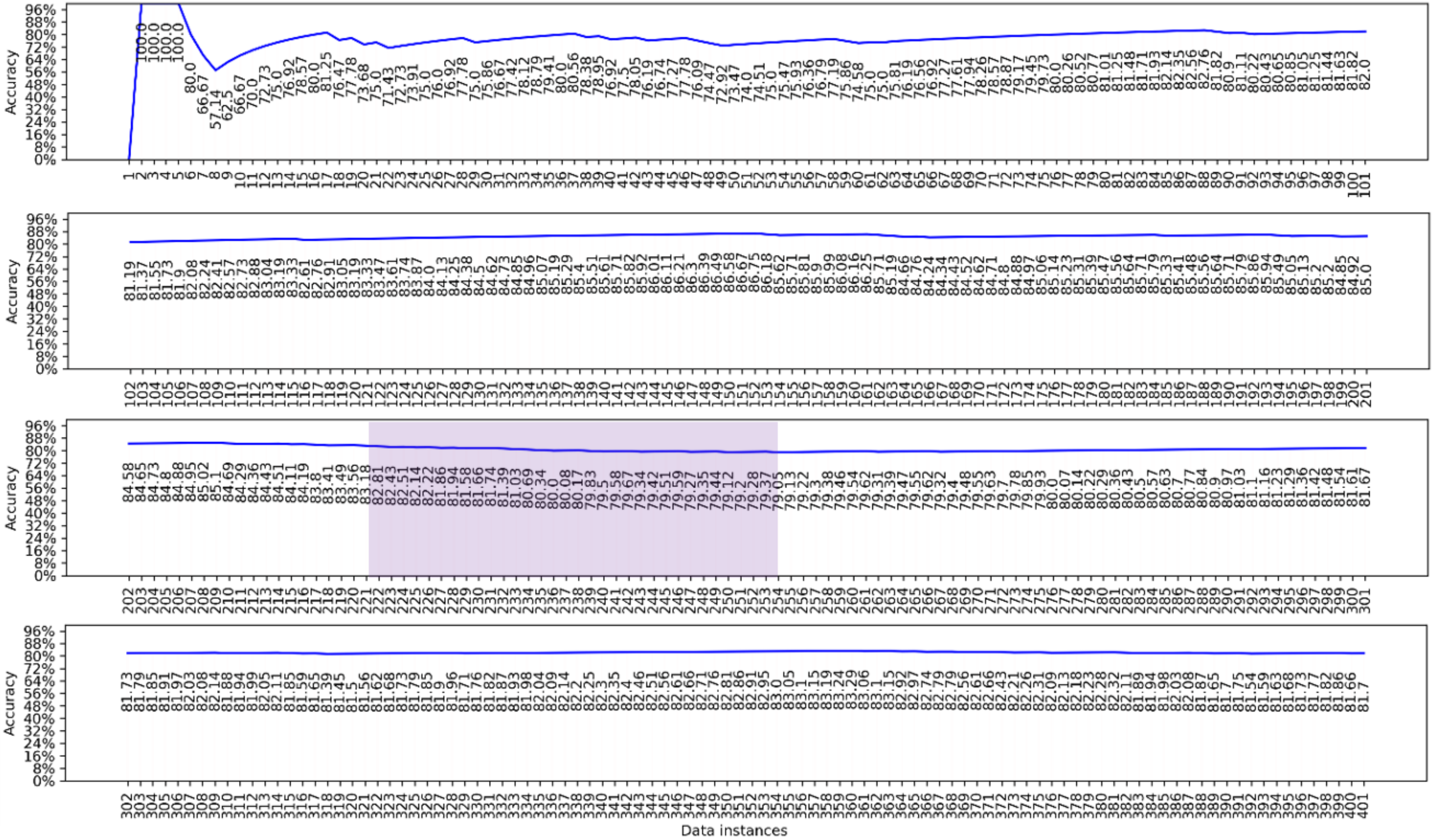


Figure A.5.: Incremental learning in the Hanover dataset (figure continues on the next page).

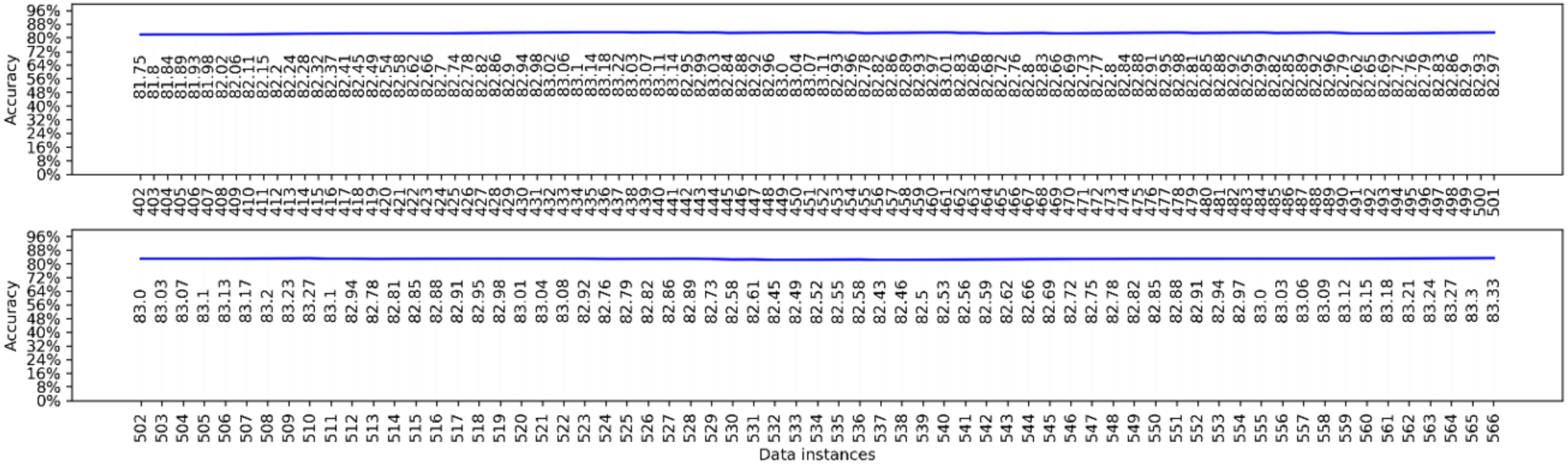


Figure A.6.: Incremental learning in the Hanover dataset (continuing from the previous figure).

List of Figures

1.1. Trajectories from moving objects in different movement modes: (a) walking, (b) cycling, (c) car driving and (d) sailing. 3
1.2. (a) Google maps eco-friendly routing. (b) Eco-routing is still not available to everyone[1]. 5
1.3. Some examples of traffic signs. 5
1.4. Intersections for which OSM contains traffic regulation relevant information (in green circles). For all other intersections no such information is present. Accessed on 04.08.2022. 7
1.5. Google Street View coverage (blue lines). The images were accessed on 09.08.2022. 8
1.6. Coverage of Mapillary Street level Imagery (green lines). The images were accessed on 10.08.2022. 9
1.7. Both images show intersections controlled by traffic signals. Successive samples during the activation of the turn signal/blinker and brake are indicated in red and yellow respectively. The spatial trajectory of the moving vehicle is indicated in blue. 11
1.8. This figure shows the recorded routes of two different vehicles in an area. Blue trajectories denote the routes. Blinker and brake occurrences are shown in red and yellow, respectively. With green and purple circles we denote clusters of similar spatial behavior. Sequences of blinker and brake followed by change of direction indicate turn pattern (green circle) and sequences of braking until stopping indicate stop maneuver (purple circle). Black dotted circles denote brake instances being observed at a single trajectory (not all the drivers that pass from this location brake, so this behavior cannot be considered as pattern). 12
1.9. Schematic illustration of the main objective of this thesis: enriching maps with traffic regulators from low cost data such as GPS trajectories. 13

2.1. Vehicle conflict points at a T-intersection: ◑ merging, ○ crossing, ● and diverging points. 17
2.2. Different types of intersections. 18
2.3. Images from intersections controlled by different regulations[2]. 21
2.4. The GPS segments. 23
2.5. Two trajectories that represent the same traveled path, sampled with different sampling periods (blue with T period and orange with $3T$ period). 25
2.6. Type of GPS data that can be stored in a GPX file: waypoints, routes and tracks. 26
2.7. A sample of the content of a GPX file, containing a track consisted of two trackpoints, with sampling period of 1 second. 26
2.8. Four movement patterns identified in the spatiotemporal data (trajectories) of four moving objects: interesting location, meeting point, moving clusters (flock) and periodic behaviour. 28
2.9. The process of spatiotemporal data mining (Sharma et al., 2022). 31
2.10. The framework for semantic trajectory mining (Alvares et al., 2009). 32

2.11. Enriching trajectories with semantic information based on stop and move detections. . . . 33
2.12. Two trajectories T_1 and T_2 with stop episodes S_1 and S_2 on the same location (blue). . . . 37
2.13. A single trajectory with starting point p_1 and ending point p_n. The region which the trajectory is observed on, includes four candidate stops C_1, C_2, C_3, C_4 (marked in blue), at two of which the moving object stops (S_3 stops at C_2 and S_4 at C_3). CB-SMoT can also detect unknown stopping locations, such as S_1 and S_2 (marked in pink). Although C_1 is crossed by the trajectory, no stopping episode is detected at it, as the requirement for staying more that Δ_{c_1} is not met. . . . 37
2.14. Hotspots where the speed of vehicles is less than 10 kmh (red color). In blue are depicted the trajectory samples. . . . 39
2.15. Speed profile of a vehicle before crossing a traffic light controlled intersection. Both graphs correspond to the same vehicle trajectory. (a) shows the speed of the last 60 meters, while (b) shows the speed of the last 20 seconds, both measured from the centre of the intersection. . . . 39
2.16. Implementation steps for solving a problem with machine learning. . . . 41
2.17. A decision tree that decides whether someone goes to work or stays at home. The binary features (true, false) are is_weekend , pending_urgent_task, is_holiday. Someone goes to work unless it is a weekend and there isn't any pending urgent task at work, or it is a weekday and a holiday and no urgent work is pending at work. . . . 44
2.18. An example of a binary decision tree on the modified Iris dataset (2 classes) for different (max) values of depth: (a-b) 1, (c-d) 2 and (e-f) 10. . . . 46
2.19. Voting classifiers. Class 1: (90% + 30% + 45% + 40% + 80%)/5 = 57%. Class 2: (10% + 70% + 55% + 60% + 20%)/5 = 43%. . . . 47
2.20. A random forest (lower right figure) created from eight decision trees. Targets from different classes are indicated in different color and shape (balls and asterisks). Decision boundaries are illustrated in yellow and orange color. . 49
2.21. The sequential process of building a gradient boosted decision tree model. New predictors are added iteratively, by training each new tree on the residual errors of the predecessor tree. . . . 50
2.22. The evolution of a single decision tree to random forest, gradient boosted decision trees and the recent implementation of the latter for faster predictions (XGBoost library). . . . 50
2.23. An example of the K-means clustering algorithm, where data instances are assigned to three clusters (pink, yellow, blue), within three iterations of point assignment and centroid update. The centroids of clusters are indicated by asterisks. . . . 54
2.24. The DBSCAN algorithm for different values of parameters. . . . 55
2.25. Core, border and noise points (min_samples = 10). . . . 55
2.26. The pool-based active learning process. . . . 60
2.27. (a) Batch learning, (b) Incremental (online) learning. . . . 61

3.1. Taxonomy of methods for traffic regulation recognition from GPS data. . . . 65
3.2. Intersection arms of (a) a three-way and (b) a four-way intersection. . . . 68

4.1. The three datasets used in this study. . . . 81
4.2. Examples of groundtruth maps. . . . 82

4.3. Tools for constructing groundtruth maps of intersection regulations. 83
4.4. Stopping (red) and deceleration (yellow) episodes detected in one trip (from east to west). The numbers in blue indicate the vehicle's speed in kmh. . . . 84
4.5. Stopping episodes (in red) detected in the Chicago dataset. 85
4.6. Illustration of the distance-related features (end-to-end distance, semi-distance and closest distance) of the static method that represent the north-south intersection arm (indicated in grey color) of a four-way intersection (in yellow). 87
4.7. The four movement patterns that describe a vehicle's crossing of an intersection arm. 88
4.8. 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 square), (d) stop more than once (here two stop events are depicted with two red squares). 89
4.9. Each intersection consists of intersection arms that connect it to nearby intersections. Classification features are calculated per arm, within half the distance of the road segment connecting the current arm to the previous arm visited by the trajectory (red dashed arrows in (a)). For each trajectory in (b) that crosses the intersection arm *j-arm* from west to east, stopping and deceleration episodes are detected within the orange indicated area along the *j-arm* in (b). 90
4.10. The reference angle system for describing the intersections arms. 93
4.11. Confusion matrices and false/true positive rates for the three datasets. 95

5.1. Traffic regulation combinations in intersections: (a)-(h) valid in Champaign and Chicago, (i)-(p) valid in Hanover, (q) an example of invalid regulation combination. 100
5.2. Confusion matrices and false/true positive rates for the three datasets. 104
5.3. 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). 107
5.4. 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. 110
5.5. Two examples of misclassified intersection regulations from the Champaign dataset. With blue diamond is depicted an incorrectly predicted regulation. A red point depicts a correctly predicted regulation. The red cross depicts the intersection center. The label of the predicted regulations contains the following information: id number of the intersection arm/ predicted label (only for the wrongly predicted regulations)/actual label/ predicted probability of label 0/ predicted probability of label 1/ predicted probability of label 2. . . 111
5.6. An example of a misclassified regulation, depicted as blue diamond, which is predicted as TS (pred:2) with probability 0.75, while its actual label is SS (act:1). This regulation is recoverable due to the information from the predicted labels of its context arms, which are predicted as SS, with probabilities 1.0 and 0.89. Both probabilities are considered high, according to the defined threshold, and because the arms are perpendicular to each other, it can be inferred that the intersection is all-way stop-controlled. The regulation with id 419 is then corrected to SS. 113

5.7. An example of an incorrectly predicted regulation (id:1323) from the Hanover dataset which is non recoverable, due to the fact that both context arms are predicted with high probabilities (arm 1323 with 0.96 and arm 1322 with 0.90). The label of the predicted regulations contains the following information: id number of the intersection arm/ predicted label (only for the wrongly predicted regulations)/actual label/ predicted probability of label 0/ predicted probability of label 1/ predicted probability of label 2. 115
5.8. Confusion matrices and false/true positive rates for the three datasets after applying consistency checks using domain knowledge rules. 116
5.9. Confusion matrices before checking for consistency the predicted labels using domain knowledge rules (a, d, g), after recovering incorrect predictions with the usage of domain knowledge rules (b, e, h), and after both correcting incorrect predicted labels and inferring regulation for context arms with no available trajectory data (c, f, i). 117

6.1. Confusion matrices and false/true positive rates for the three datasets after applying K-means clustering. 122
6.2. Predicted labels vs. actual labels. The x-axis corresponds to the average crossing speed and y-axis to the duration of last stopping episode. In Chicago, green corresponds to SS, blue corresponds to TS and red to UN. In Hanover, green corresponds to PS, blue to TS and red to UN. 123
6.3. Settings of self-training experiments using different sizes of labeled/unlabeled data. 124
6.4. Classification performance using self-training under different amounts of labeled/unlabeled training data. The percentages of data used as training data of x-axis refer to labeled data. 125
6.5. Active learning framework for traffic regulation recognition. 126
6.6. Settings of active-learning experiments with different sizes of actively labeled training data. Successive experiments query a larger amount of labels from a hypothetical human domain expert (annotator), as explained in Section 2.3.5. 126
6.7. Classification performance using active learning under different amounts of (actively) labeled training data. 127
6.8. Classification performance of cluster-then-label and cluster-then-label-propagation under different amounts of labeled training data. 128
6.9. Classification performance of cluster-then-label, cluster-then-label-propagation and baseline model under different amounts of labeled training data. 129
6.10. Experiments on transferability of learning are applied as illustrated in the right figure. A TRR classifier is trained on data from a city A and then applied to data from a city B. The classification accuracy is denoted as Ac. If training and predicting are performed on the same dataset, as illustrated in the left figure, then the accuracy is denoted as Ac_{orig}. 129
6.11. Confusion matrices of the hybrid all-static model for the six experiments. . . 130
6.12. Examples of uncontrolled intersections in Hanover (all-way UN) and in Champaign (UN-SS). In red are framed stop signs and a stop line. 135
6.13. Confusion matrices of the dynamic one-arm model for Exp3, Exp4, Exp5, and Exp6. 136
6.14. A TRR incremental learning scenario. 138
6.15. Classification performance in the three datasets under a TRR incremental learning scenario. 139

6.16. Data instances of the Hanover dataset in order of arrival (1 is the first instance of the stream to arrive, 2 is the second, etc.): from 1 to 220 data instances (green spheres), 221-254 (red spheres) and 255-566 (blue spheres). The circled area shown in (c) corresponds to the dotted circled areal indicated in (a). . 140

A.1. Feature importance. 154
A.2. Incremental learning in the Champaign dataset (figure continues on the next page). 155
A.3. Incremental learning in the Champaign dataset (continuing from the previous figure). 156
A.4. Incremental learning in the Chicago dataset. 157
A.5. Incremental learning in the Hanover dataset (figure continues on the next page). 158
A.6. Incremental learning in the Hanover dataset (continuing from the previous figure). 159

List of Tables

2.1. Intersection design considerations (Design-Manual, 2021). 19
2.2. A confusion matrix of a binary classification problem. 51

3.1. Reviewed articles in chronological order. 66
3.2. Extracted information from the reviewed articles. 67

4.1. Datasets used for testing the proposed methods. 80
4.2. Overview of the classification features derived for the dynamic TRR approach. 92
4.3. Classification accuracy (Acc) and F1-score (F_1) of the four TRR models. The highlighted number(s) per dataset correspond(s) to the best accuracy achieved in the respective dataset. 94
4.4. Classification results of the hybrid approach (GB). 94

5.1. Combinations of traffic regulators at intersections in the three datasets. The numbers in the cells in the first block of the table refer to number of intersections that belongs to a certain regulation combination (e.g. UN-UN-SS, PS-PS-YS, etc). In the second block the numbers refer to the percentage of intersections that belongs to a certain regulation combination out of the total number of intersections (e.g. Champ. 84% = 293/350). 100
5.2. Classification accuracy (Acc) and F1-score (F_1) of the TRR models. Highlighted are the best one-arm and all-arm models for each dataset. 103
5.3. Classification performance of the hybrid all-static model (GB). 104
5.4. Classification accuracy of the TRR methods under different sampling rates (undersampling). The original datasets are highlighted in grey. 105
5.5. Classification performance of the reduced and default models. 106
5.6. Dataset size for different trajectory direction settings with minimum number of trajectories per intersection arm equal to 5 (Champaign dataset). 108
5.7. The wrong predictions in the Champaign dataset (Rec.: whether the regulation is recoverable or not, Num.Adj.: number of arms from the same intersection having predicted labels (context arms). 112
5.8. The wrong predictions in the Chicago dataset. (Rec.: whether the regulation is recoverable or not, Num.Adj.: number of arms from the same intersection having predicted labels (context arms). 113
5.9. The wrong predictions in the Hanover dataset. (Rec.: whether the regulation is recoverable or not, Num.Adj.: number of arms from the same intersection having predicted labels (context arms). 114
5.10. Classification results of the default model before and after applying domain knowledge rules for recovering incorrect predictions and predicting regulation labels, when possible, for arms with no available trajectory data. 115

6.1. Mean values of the duration of the last stop episode and of the average crossing speed, calculated from the data assigned to each cluster. 122
6.2. Classification performance after matching clusters to regulator labels. 122

6.3. Classification results of the six experiments on the transferability of learning between cities. Ac: accuracy on the Predict dataset when training is done with data from the city given in the Training dataset column; $\Delta_{Ac} = Ac_{Orig} - Ac$, where Ac_{Orig} is the accuracy when training is done in the same city as the Predict dataset (Ac_{Orig} is given in the lower block of rows of the table). The best accuracy per experiment across the different TRR methods is indicated in bold. 130
6.4. Combinations of traffic regulators at intersections in the three datasets. The numbers refer to number of intersections. 133
6.5. Combinations of traffic regulators at the intersections of the three datasets, where intersection arms are crossed by at least five straight trajectories. The numbers refer to number of intersections. 133
6.6. Classification results of the six experiments on the transferability of learning between cities (hybrid all-static approach). Ac: accuracy on the Predict dataset when training is done with data from the city given in the Training dataset column; $\Delta_{Ac} = Ac_{Orig} - Ac$, where Ac_{Orig} is the accuracy when training is done in the same city as the Predict dataset. The results under the "original" columns refer to the experiments which use the GPS data with their original sampling rate. The "undersampled" results, refer to the experiments where the Champaign and Hanover datasets are (under)sampled (see page 131, observation 5, for the settings of undersampling). The best accuracy per experiment is indicated in bold. 136

A.1. Classification results of the hybrid one-arm model (GB). 151
A.2. Classification results of the hybrid all-static model (tuned GB). 151
A.3. Classification results of the reduced and default models. 152
A.4. Classification results of the hybrid all-static model after applying consistency check with domain knowledge rules. 153
A.5. Clustering results (K-means) of the hybrid all-static model. 153

Bibliography

Abreu, F. H. O., Soares, A., Paulovich, F. V., Matwin, S., 2021. A trajectory scoring tool for local anomaly detection in maritime traffic using visual analytics. *ISPRS International Journal of Geo-Information* 10 (6).

Agamennoni, G., Nieto, J., Nebot, E., 2009. Mining gps data for extracting significant places. In: *Proceedings of the IEEE International Conference on Robotics and Automation.* pp. 855–862.

Ahmed, M., Karagiorgou, S., Pfoser, D., Wenk, C., 2015. A comparison and evaluation of map construction algorithms using vehicle tracking data. *GeoInformatica* 19 (3), pp. 601–632.

Alshayeb, S., Stevanovic, A., Effinger, J. R., 2021. Investigating impacts of various operational conditions on fuel consumption and stop penalty at signalized intersections. *International Journal of Transportation Science and Technology.*

Alvares, L. O., Bogorny, V., Kuijpers, B., de Macedo, J. A. F., Moelans, B., Vaisman, A., 2007. A model for enriching trajectories with semantic geographical information. In: *Proceedings of the 15th Annual ACM International Symposium on Advances in Geographic Information Systems.* GIS '07. ACM, New York, NY, USA, pp. 22:1–22:8.

Alvares, L. O., Oliveira, G., Heuser, C. A., Bogorny, V., 2009. A framework for trajectory data preprocessing for data mining. In: *SEKE.* Knowledge Systems Institute Graduate School, pp. 698–702.

Aly, H., Basalamah, A., Youssef, M., 2017. Automatic rich map semantics identification through smartphone-based crowd-sensing. *IEEE Transactions on Mobile Computing* 16 (10), pp. 2712–2725.

Andersen, O., Torp, K., 2017. Sampling frequency effects on trajectory routes and road network travel time. In: *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems.* SIGSPATIAL '17. Association for Computing Machinery, New York, NY, USA.

Andrzejewski, W., Boinski, P., 2021. Maximal mixed-drove co-occurrence patterns. In: Bellatreche, L., Dumas, M., Karras, P., Matulevičius, R. (Hrsg.), *Advances in Databases and Information Systems.* Springer International Publishing, Cham, pp. 15–29.

Ardianto, S., Chen, C., Hang, H., 2017. Real-time traffic sign recognition using color segmentation and SVM. In: *Proceedings of the International Conference on Systems, Signals and Image Processing (IWSSIP).* pp. 1–5.

Ashbrook, D., Starner, T., 2003. Using gps to learn significant locations and predict movement across multiple users. *Personal Ubiquitous Computing* 7 (5), pp. 275–286.

Atev, S., Masoud, O., Papanikolopoulos, N., 2006. Learning traffic patterns at intersections by spectral clustering of motion trajectories. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems.* pp. 4851–4856.

Balali, V., Golparvar-Fard, M., 2016. Evaluation of multiclass traffic sign detection and classification methods for U.S. roadway asset inventory management. *Journal of Computing in Civil Engineering* 30 (2), pp. 04015022.

Beitel, D., Stipancic, J., Manaugh, K., Miranda-Moreno, L., 2018. Assessing safety of shared space using cyclist-pedestrian interactions and automated video conflict analysis. *Transportation Research Part D: Transport and Environment* 65, pp. 710–724.

Bermingham, L., Lee, I., 2018. A probabilistic stop and move classifier for noisy gps trajectories. *Data Mining and Knowledge Discovery* 32 (6), pp. 1634–1662.

Biagioni, J., Eriksson, J., 2012. Inferring road maps from global positioning system traces. *Transportation Research Record: Journal of the Transportation Research Board* 2291, pp. 61–71.

Bifet, A., Holmes, G., Pfahringer, B., 2010. Leveraging bagging for evolving data streams. In: Balcázar, J. L., Bonchi, F., Gionis, A., Sebag, M. (Hrsg.), *Machine Learning and Knowledge Discovery in Databases.* Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 135–150.

Birant, D., Kut, A., 2007. ST-DBSCAN: An algorithm for clustering spatial-temporal data. *Data and Knowledge Engineering* 60 (1), pp. 208–221.

Butt, U. M., Letchmunan, S., Hassan, F. H., Ali, M., Baqir, A., Sherazi, H. H. R., 2020. Spatio-temporal crime hotspot detection and prediction: a systematic literature review. *IEEE Access* 8, pp. 166553–166574.

Cao, L., Krumm, J., 2009. From gps traces to a routable road map. In: *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems.* GIS '09. ACM, New York, NY, USA, pp. 3–12.

Carbonara, L., Borrowman, A., 1998. A comparison of batch and incremental supervised learning algorithms. In: Żytkow, J. M., Quafafou, M. (Hrsg.), *Principles of Data Mining and Knowledge Discovery.* Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 264–272.

Carisi, R., Giordano, E., Pau, G., Gerla, M., 2011. Enhancing in vehicle digital maps via gps crowdsourcing. In: *Proceedings of the Eighth International Conference on Wireless On-Demand Network Systems and Services.* pp. 27–34.

Chapelle, O., Scholkopf, B., Zien, A., 2006. Semi-supervised learning. 2006. *Cambridge, Massachusettes: The MIT Press View Article* 2.

Chen, M., Yang, J., Hu, L., Hossain, M. S., Muhammad, G., 2018. Urban healthcare big data system based on crowdsourced and cloud-based air quality indicators. *IEEE Communications Magazine* 56 (11), pp. 14–20.

Chen, T., Guestrin, C., 2016. XGBoost: A scalable tree boosting system. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* KDD '16. ACM, New York, NY, USA, pp. 785–794.

Cheng, H., Lei, H., Zourlidou, S., Sester, M., 2022. Traffic control recognition with an attention mechanism using speed-profiles and satellite imagery data. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 43, pp. 287–293.

Comito, C., Falcone, D., Talia, D., 2016. Mining human mobility patterns from social geo-tagged data. *Pervasive and Mobile Computing* 33, pp. 91–107.

Dang, V.-C., Kubo, M., Sato, H., Yamaguchi, A., Namatame, A., 2014. A simple braking model for detecting incidents locations by smartphones. In: *Proceedings of the Seventh IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA).* pp. 1–5.

Danka, T., Horvath, P., 2018. modAL: A modular active learning framework for Python. Available on arXiv at `https://arxiv.org/abs/1805.00979`.

Davies, J. J., Beresford, A. R., Hopper, A., 2006. Scalable, distributed, real-time map generation. *IEEE Pervasive Computing* 5 (4), pp. 47–54.

De Groeve, J., Van de Weghe, N., Ranc, N., Neutens, T., Ometto, L., Rota-Stabelli, O., Cagnacci, F., 2016. Extracting spatio-temporal patterns in animal trajectories: an ecological application of sequence analysis methods. *Methods in Ecology and Evolution* 7 (3), pp. 369–379.

Depauw, L., Blondeel, H., De Lombaerde, E., De Pauw, K., Landuyt, D., Lorer, E., Vangansbeke, P., Vanneste, T., Verheyen, K., De Frenne, P., 2022. The use of photos to investigate ecological change. *Journal of Ecology* 110 (6), pp. 1220–1236.

Design-Manual, 2021. Design Manual. Washington State, Department of Transportation. URL `https://wsdot.wa.gov/publications/manuals/fulltext/M22-01/1300.pdf`

Efentakis, A., Grivas, N., Pfoser, D., Vassiliou, Y., 2017. Crowdsourcing turning-restrictions from map-matched trajectories. *Information Systems* 64, pp. 221–236.

El-Rabbany, A., 2002. Introduction to GPS. Artech House, Norwood, MA, 02062.

Eriksson, J., Girod, L., Hull, B., Newton, R., Madden, S., Balakrishnan, H., 2008. The pothole patrol: using a mobile sensor network for road surface monitoring. In: *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*. MobiSys '08. ACM, New York, NY, USA, pp. 29–39.

Ester, M., Kriegel, H. P., Sander, J., Xu, X., 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD '96)*. pp. 226–231.

Fathi, A., Krumm, J., 2010. Detecting road intersections from gps traces. In: Fabrikant, S., Reichenbacher, T., van Kreveld, M., Schlieder, C. (Hrsg.), *Geographic Information Science*. vol. 6292 from Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 56–69.

Feng, Z., Zhu, Y., 2016. A survey on trajectory data mining: techniques and applications. *IEEE Access* 4, pp. 2056–2067.

Feuerhake, U., 2016. Recognition of repetitive movement patterns - the case of football analysis. *ISPRS International Journal of Geo-Information* 5 (11).

Fitzner, D., Sester, M., 2016. Field motion estimation with a geosensor network. *ISPRS International Journal of Geo-Information* 5 (10).

Fox, A., Kumar, B. V., Chen, J., Bai, F., 2017. Multi-lane pothole detection from crowdsourced undersampled vehicle sensor data. *IEEE Transactions on Mobile Computing* 16 (12), pp. 3417–3430.

Ganti, R. K., Pham, N., Ahmadi, H., Nangia, S., Abdelzaher, T. F., 2010. GreenGPS: a participatory sensing fuel-efficient maps application. In: *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*. MobiSys '10. ACM, New York, NY, USA, pp. 151–164.

Gastaldi, M., Meneguzzer, C., Rossi, R., Lucia, L. D., Gecchele, G., 2014. Evaluation of air pollution impacts of a signal control to roundabout conversion using microsimulation. *Transportation Research Procedia* 3, pp. 1031–1040, 17th Meeting of the EURO Working Group on Transportation, EWGT2014, 2-4 July 2014, Sevilla, Spain.

Géron, A., 2019. Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent system (2nd ed.). O'Reilly.

Giannotti, F., Nanni, M., Pinelli, F., Pedreschi, D., 2007. Trajectory pattern mining. In: *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '07. ACM, New York, NY, USA, pp. 330–339.

Given, L. M., 2008. In the SAGE encyclopedia of qualitative research methods. SAGE Publications, Ch. Naturalistic data, pp. 547–547.

Goldberg, A., Zhu, X., Singh, A., Xu, Z., Nowak, R., 2009. Multi-manifold semi-supervised learning. In: van Dyk, D., Welling, M. (Hrsg.), *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*. vol. 5. PMLR, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, pp. 169–176.

Golze, J., Zourlidou, S., Sester, M., 2020. Traffic regulator detection using gps trajectories. *KN - Journal of Cartography and Geographic Information.*

Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfharinger, B., Holmes, G., Abdessalem, T., 2017. Adaptive random forests for evolving data stream classification. *Machine Learning* 106 (9-10), pp. 1469–1495.

Gong, L., Liu, X., Wu, L., Liu, Y., 2016. Inferring trip purposes and uncovering travel patterns from taxi trajectory data. *Cartography and Geographic Information Science* 43 (2), pp. 103–114.

Goodchild, M. F., Fu, P., Rich, P., 2007. Sharing geographic information: an assessment of the Geospatial One-Stop. *Annals of the Association of American Geographers* 97 (2), pp. 250–266.

Google Maps, 2022. Google Maps: eco-routing. `https://www.gstatic.com/gumdrop/sustainability/google-maps-eco-friendly-routing.pdf`, accessed: 2022-08-12.

Google Street View, 2022. Google Street View: street view imagery. `https://www.google.com/streetview/`, accessed: 2022-08-08.

Gudmundsson, J., Laube, P., Wolle, T., 2008. Movement patterns in spatio-temporal data. *Encyclopedia of GIS* 726, pp. 732.

Gummidi, S. R. B., Xie, X., Pedersen, T. B., 2019. A survey of spatial crowdsourcing. *ACM Transactions on Database Systems.* 44 (2).

Guo, B., Yu, Z., Zhang, D., Zhou, X., 2014. From participatory sensing to mobile crowd sensing. *CoRR* abs/1401.3090.

Guo, D., 2008. Mining traffic condition from trajectories. In: *Proceedings of the Fifth International Conference on Fuzzy Systems and Knowledge Discovery, 2008. FSKD '08.* vol. 4. pp. 256–260.

Harris, P., Comber, A., Tsutsumida, N., 2017. Specifying regression models for spatio-temporal data sets.

He, L., Niu, X., Chen, T., Mei, K., Li, M., 2022. Spatio-temporal trajectory anomaly detection based on common sub-sequence. *Applied Intelligence* 52 (7), pp. 7599–7621.

He, S., Bastani, F., Abbar, S., Alizadeh, M., Balakrishnan, H., Chawla, S., Madden, S., 2018. RoadRunner: improving the precision of road network inference from gps trajectories. In: *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems.* SIGSPATIAL '18. ACM, New York, NY, USA, pp. 3–12.

Heipke, C., 2010. Crowdsourcing geospatial data. *ISPRS Journal of Photogrammetry and Remote Sensing* 65 (6), pp. 550–557.

Hong, S., Vatsavai, R. R., 2016. A scalable probabilistic change detection algorithm for very high resolution (VHR) satellite imagery. In: *Proceedings of the IEEE International Congress on Big Data (BigData Congress).* pp. 275–282.

Hu, H., Li, G., Bao, Z., Cui, Y., Feng, J., 2016. Crowdsourcing-based real-time urban traffic speed estimation: from trends to speeds. In: *Proceedings of the 32nd IEEE International Conference on Data Engineering (ICDE).* pp. 883–894.

Hu, S., Su, L., Liu, H., Wang, H., Abdelzaher, T. F., 2015. SmartRoad: smartphone-based crowd sensing for traffic regulator detection and identification. *ACM Transactions on Sensor Networks* 11 (4), pp. 55:1–55:27.

Hu, W., Xie, D., Tan, T., 2004. A hierarchical self-organizing approach for learning the patterns of motion trajectories. *IEEE Transactions on Neural Networks* 15 (1), pp. 135–144.

Huang, H., Zhang, L., Sester, M., 2014. A recursive Bayesian filter for anomalous behavior detection in trajectory data. Springer International Publishing, Cham, pp. 91–104.

Huang, S., Lin, H., Chang, C., 2017. An in-car camera system for traffic sign detection and recognition. In: *Proceedings of the Joint 17th World Congress of International Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA-SCIS).* pp. 1–6.

Jiang, F., Tsaftaris, S., Wu, Y., Katsaggelos, A., 2009. Detecting anomalous trajectories from highway traffic data. `http://www.ccitt.northwestern.edu/documents/2009.Jiang_Tsaftaris_Wu_Katsaggelos_pub.pdf`, accessed: 2015-12-05.

Kalinic, M., Jukka, M., 2018. Kernel density estimation (KDE) vs. hot-spot analysis-detecting criminal hot spots in the city of San Francisco. In: *Accepted Papers from the 21st AGILE Conference on Geo-Information Science, 12-15 June, Lund, Sweden.*

Karachiwalla, R., Pinkow, F., 2021. Understanding crowdsourcing projects: a review on the key design elements of a crowdsourcing initiative. *Creativity and Innovation Management* 30 (3), pp. 563–584.

Kawale, J., Chatterjee, S., Ormsby, D., Steinhaeuser, K., Liess, S., Kumar, V., 2012. Testing the significance of spatio-temporal teleconnection patterns. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

Kosonen, M., Henttonen, K., 2015. Cheer the crowd? Facilitating user participation in idea crowdsourcing. *International Journal of Technology Marketing* 10 (1), pp. 95–110.

Krisp, J. M., Keler, A., 2015. Car navigation - computing routes that avoid complicated crossings. *International Journal of Geographical Information Science* 0 (0), pp. 1–13.

Lam, H. T., 2016. A concise summary of spatial anomalies and its application in efficient real-time driving behaviour monitoring. In: *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems.* Association for Computing Machinery, New York, NY, USA.

Lane, N. D., Eisenman, S. B., Musolesi, M., Miluzzo, E., Campbell, A. T., 2008. Urban sensing systems: opportunistic or participatory. In: *Proceedings of the 9th ACM Workshop on Mobile Computing Systems and Applications (HOTMOBILE '08).*

Laube, P., 2009. Volume 3: Behaviour Monitoring and Interpretation - BMI. Ch. Progress in movement pattern analysis, pp. 43–71.

Laureshyn, A., Aström, K., Brundell-Freij, K., 2009. From speed profile data to analysis of behavior classification by pattern recognition techniques. {*IATSS*} *Research* 33 (2), pp. 88–98.

Lee, S., Lee, C., Mun, K. G., Kim, D., 2022. Decision tree algorithm considering distances between classes. *IEEE Access* 10, pp. 69750–69756.

Lefèvre, S., Guzman, J. I., Laugier, C., 2011. Context-based estimation of driver intent at road intersections. In: *Proceedings of the IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems (CIVTS).* pp. 67–72.

Lefèvre, S., Laugier, C., Ibañez-Guzmàn, J., 2012. Risk assessment at road intersections: comparing intention and expectation. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV).* pp. 165–171.

Li, J., Qin, Q., Han, J., Tang, L.-A., Lei, K. H., 2015. Mining trajectory data and geotagged data in social media for road map inference. *Transactions in GIS* 19 (1), pp. 1–18.

Lian, J., Zhang, L., 2018. One-month Beijing taxi gps trajectory dataset with taxi ids and vehicle status. In: *Proceedings of the First Workshop on Data Acquisition To Analysis.* DATA '18. Association for Computing Machinery, New York, NY, USA, pp. 3â4.

Liao, Z., Xiao, H., Liu, S., Liu, Y., Yi, A., 2021. Impact assessing of traffic lights via gps vehicle trajectories. *ISPRS International Journal of Geo-Information* 10 (11).

Lughofer, E., 2012. Hybrid active learning for reducing the annotation effort of operators in classification systems. *Pattern Recognition* 45 (2), pp. 884–896.

Maciag, P. S., Kryszkiewicz, M., Bembenik, R., 2019. Discovery of closed spatio-temporal sequential patterns from event data. In: *Procedia Computer Science.* vol. 159. pp. 707–716.

Makris, D., Ellis, T., 2002. Spatial and probabilistic modelling of pedestrian behaviour. In: *Proceedings of the British Machine Vision Conference 2002, BMVC 2002, Cardiff, UK, 2-5 September 2002.* pp. 1–10.

Makris, D., Ellis, T., 2003. Automatic learning of an activity-based semantic scene model. In: *Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance.* pp. 183–188.

Makris, D., Ellis, T., 2005. Learning semantic scene models from observing activity in visual surveillance. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 35 (3), pp. 397–408.

Mapillary, 2022. Mapillary: a street-level imagery platform. `https://www.mapillary.com/`, accessed: 2022-04-20.

Mapscape, 2022. Incremental updating. `http://www.mapscape.eu/telematics/incremental-updating.html`, accessed: 2022-08-08.

Mariescu-Istodor, R., Fränti, P., 2018. CellNet: inferring road networks from gps trajectories. *ACM Transactions on Spatial Algorithms and Systems* 4 (3), pp. 8:1–8:22.

Mathur, S., Jin, T., Kasturirangan, N., Chandrashekharan, J., Xue, W., Gruteser, M., Trappe, W., 2010. ParkNet: Drive-by sensing of road-side parking statistics. In: *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys'10).* pp. 123–136.

Méneroux, Y., Guilcher, A., Saint Pierre, G., Hamed, M., Mustiere, S., Orfila, O., 2020. Traffic signal detection from in-vehicle gps speed profiles using functional data analysis and machine learning. *International Journal of Data Science and Analytics* 10, pp. 101–119.

Merry, K., Bettinger, P., 2019. Smartphone gps accuracy study in an urban environment. *PLoS One* 14 (7).

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., 2015. Crowdsourced earthquake early warning. *Science Advances* 1 (3).

Mitchell, T. M., 1997. Machine Learning, 1. Edition. McGraw-Hill, Inc., USA.

Mohan, P., Padmanabhan, V. N., Ramjee, R., 2008. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In: *Proceedings of the 6th ACM conference on Embedded network sensor systems.* pp. 323–336.

Montiel, J., Halford, M., Mastelini, S. M., Bolmier, G., Sourty, R., Vaysse, R., Zouitine, A., Gomes, H. M., Read, J., Abdessalem, T., Bifet, A., 2020. River: machine learning for streaming data in Python. *CoRR* abs/2012.04740.

Morris, B., Trivedi, M., 2009. Learning trajectory patterns by clustering: experimental studies and comparative evaluation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009.* pp. 312–319.

Mou, L., Bruzzone, L., Zhu, X. X., 2019. Learning spectral-spatial-temporal features via a recurrent convolutional neural network for change detection in multispectral imagery. *IEEE Transactions on Geoscience and Remote Sensing* 57 (2), pp. 924–935.

Muckell, J., Olsen, P. W., Hwang, J.-H., Lawson, C. T., Ravi, S. S., 2014. Compression of trajectory data: a comprehensive evaluation and new approach. *GeoInformatica* 18 (3), pp. 435–460.

Muller, C., Chapman, L., Johnston, S., Kidd, C., Illingworth, S., Foody, G., Overeem, A., Leigh, R., 2015. Crowdsourcing for climate and atmospheric sciences: current status and future potential. *International Journal of Climatology* 35 (11), pp. 3185–3203.

Munoz-Organero, M., Ruiz-Blaquez, R., Sánchez-Fernández, L., 2018. Automatic detection of traffic lights, street crossings and urban roundabouts combining outlier detection and deep learning classification techniques based on gps traces while driving. *Computers, Environment and Urban Systems* 68, pp. 1–8.

Niehöfer, B., Burda, R., Wietfeld, C., Bauer, F., Lueert, O., 2009. GPS community map generation for enhanced routing methods based on trace-collection by mobile phones. In: *Proceedings of the First International Conference on Advances in Satellite and Space Communications.* pp. 156–161.

Niu, X., Wang, S., Wu, C. Q., Li, Y., Wu, P., Zhu, J., 2021. On a clustering-based mining approach with labeled semantics for significant place discovery. *Information Sciences* 578, pp. 37–63.

Palma, A. T., Bogorny, V., Kuijpers, B., Alvares, L. O., 2008. A clustering-based approach for discovering interesting places in trajectories. In: *Proceedings of the ACM Symposium on Applied Computing.* SAC '08. ACM, New York, NY, USA, pp. 863–868.

Parent, C., Spaccapietra, S., Renso, C., Andrienko, G., Andrienko, N., Bogorny, V., Damiani, M. L., Gkoulalas-Divanis, A., Macedo, J., Pelekis, N., Theodoridis, Y., Yan, Z., 2013. Semantic trajectories modeling and analysis. *ACM Computing Surveys.* 45 (4).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al., 2011. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research* 12 (Oct), pp. 2825–2830.

Peker, A. U., Tosun, O., Akin, H. L., Acarman, T., 2014. Fusion of map matching and traffic sign recognition. In: *Proceedings of the IEEE Intelligent Vehicles Symposium.* pp. 867–872.

Phithakkitnukoon, S., Horanont, T., Di Lorenzo, G., Shibasaki, R., Ratti, C., 2010. Human behavior understanding. Springer Berlin Heidelberg, Ch. Activity-aware map: identifying human daily activity pattern using mobile phone data, pp. 14–25.

Picaut, J., Fortin, N., Bocher, E., Petit, G., Aumond, P., Guillaume, G., 2019. An open-science crowdsourcing approach for producing community noise maps using smartphones. *Building and Environment* 148, pp. 20–33.

Piciarelli, C., Foresti, G., 2005. Toward event recognition using dynamic trajectory analysis and prediction. In: *Proceedings of the IEE International Symposium on Imaging for Crime Detection and Prevention, 2005. ICDP 2005.* IET, pp. 131–134.

Pribe, C. A., Rogers, S. O., 1999. Learning to associate observed driver behavior with traffic controls. *Transportation Research Record: Journal of the Transportation Research Board* 1679 (1), pp. 95–100.

Quddus, M. A., Ochieng, W. Y., Noland, R. B., 2007. Current map-matching algorithms for transport applications: state-of-the art and future research directions. *Transportation Research Part C: Emerging Technologies* 15 (5), pp. 312–328.

Read, J., Bifet, A., Pfahringer, B., Holmes, G., 2012. Batch-incremental versus instance-incremental learning in dynamic and evolving data. In: Hollmén, J., Klawonn, F., Tucker, A. (Hrsg.), *Advances in Intelligent Data Analysis XI.* Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 313–323.

Rocha, J. A. M. R., Times, V. C., Oliveira, G., Alvares, L. O., Bogorny, V., 2010. DB-SMoT: A direction-based spatio-temporal clustering method. In: *Proceedings of the 5th IEEE International Conference in Intelligent Systems.* pp. 114–119.

Rodosthenous, C. T., Michael, L., 2021. A crowdsourcing methodology for improved geographic focus identification of news-stories. In: *ICAART (2).* pp. 680–687.

Romano, B., Jiang, Z., 2017. Visualizing traffic accident hotspots based on spatial-temporal network kernel density estimation. In: *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems.* SIGSPATIAL '17. Association for Computing Machinery, New York, NY, USA.

Salpietro, R., Bedogni, L., Di Felice, M., Bononi, L., 2015. Park Here! a smart parking system based on smartphones' embedded sensors and short range Communication Technologies. In: *Proceedings of the 2nd IEEE World Forum on Internet of Things (WF-IoT).* pp. 18–23.

Saremi, F., 2016. Participatory sensing fuel-efficient navigation system greengps. Dissertation, University of Illinois at Urbana-Champaign.

Saremi, F., Abdelzaher, T. F., 2015. Combining map-based inference and crowd-sensing for detecting traffic regulators. In: *Proceedings of the 12th IEEE International Conference on Mobile Ad Hoc and Sensor Systems*. pp. 145–153.

Satzoda, R. K., Martin, S., Ly, M. V., Gunaratne, P., Trivedi, M. M., 2013. Towards automated drive analysis: a multimodal synergistic approach. In: *Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. pp. 1912–1916.

Sester, M., Feuerhake, U., Kuntzsch, C., Zhang, L., 2012. Revealing underlying structure and behaviour from movement data. *KI - Künstliche Intelligenz* 26.

Settles, B., 2009. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.
URL `http://axon.cs.byu.edu/~martinez/classes/778/Papers/settles.activelearning.pdf`

Sharma, A., Jiang, Z., Shekhar, S., 2022. Spatiotemporal data mining: a survey. URL `https://arxiv.org/abs/2206.12753`

Shekhar, S., Jiang, Z., Ali, R. Y., Eftelioglu, E., Tang, X., Gunturi, V. M. V., Zhou, X., 2015. Spatiotemporal data mining: a computational perspective. *ISPRS International Journal of Geo-Information* 4 (4), pp. 2306–2338.

Spaccapietra, S., Parent, C., Damiani, M. L., de Macedo, J. A., Porto, F., Vangenot, C., 2008. A conceptual view on trajectories. *Data and Knowledge Engineering* 65 (1), pp. 126–146.

Spinsanti, L., Celli, F., Renso, C., 2010. Where you stop is who you are: understanding people's activities by places visited. CEUR.

Tan, P.-N., Steinbach, M., Karpatne, A., Kumar, V., 2018. Introduction to data mining, 2. Edition. Pearson.

Tang, J., Deng, M., Huang, J., Liu, H., Chen, X., 2019. An automatic method for detection and update of additive changes in road network with gps trajectory data. *ISPRS International Journal of Geo-Information* 8 (9).

Tang, L., Kan, Z., Zhang, X., Yang, X., Huang, F., Li, Q., 2016. Travel time estimation at intersections based on low-frequency spatial-temporal gps trajectory big data. *Cartography and Geographic Information Science* 43 (5), pp. 417–426.

topografix.com, 2022. Official GPX webpage: Resources. `https://www.topografix.com/gpx_resources.asp`, accessed: 2022-08-18.

U.S. Federal Aviation Administration, 2022. Satellite navigation - GPS - how it works. `https://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/navservices/gnss/gps/howitworks`, accessed: 2022-08-17.

Van Engelen, J. E., Hoos, H. H., 2020. A survey on semi-supervised learning. *Machine Learning* 109 (2), pp. 373–440.

Vij, D., Aggarwal, N., 2018. Smartphone based traffic state detection using acoustic analysis and crowdsourcing. *Applied Acoustics* 138, pp. 80–91.

Wage, O., Sester, M., 2021. Joint estimation of road roughness from crowd-sourced bicycle acceleration measurements. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* V-4-2021, pp. 89–96.

Wang, J., Wang, C., Song, X., Raghavan, V., 2017. Automatic intersection and traffic rule detection by mining motor vehicle gps trajectories. *Computers, Environment and Urban Systems* 64, pp. 19–29.

Wang, S., Niu, X., Fournier-Viger, P., Zhou, D., Min, F., 2022. A graph based approach for mining significant places in trajectory data. *Information Sciences* 609, pp. 172–194.

Wei, L., Li, Z., Gong, J., Gong, C., Li, J., 2021. Autonomous driving strategies at intersections: scenarios, state-of-the-art, and future outlooks. In: *Proceedings of the IEEE International Intelligent Transportation Systems Conference (ITSC).* pp. 44–51.

Weng, Y.-C., 2007. Spatiotemporal changes of landscape pattern in response to urbanization. *Landscape and Urban Planning* 81 (4), pp. 341–353.

Wisch, M., Hellmann, A., Lerner, M., Hierlinger, T., Labenski, V., Wagner, M., Feifel, H., Robescu, O., Renoux, P., Groult, X., 2019. Car-to-car accidents at intersections in Europe and identification of use cases for the test and assessment of respective active vehicle safety systems. In: *Proceedings of the 26th International Technical Conference on the Enhanced Safety of Vehicles (ESV), Eindhoven, 2019.*

Wu, J., Zhu, Y., Ku, T., Wang, L., 2013. Detecting road intersections from coarse-gained gps traces based on clustering. *Journal of Computers* 8 (1), pp. 2959–2965.

XGBoost Python, 2022. XGBoost Python Library. `https://xgboost.readthedocs.io/en/stable/python/index.html`, accessed: 2022-02-15.

Xiang, L., Gao, M., Wu, T., 2016. Extracting stops from noisy trajectories: a sequence oriented clustering approach. *ISPRS International Journal of Geo-Information* 5 (3).

Xiao, X., Gao, R., Xing, W., Li, C., Liu, L., 2021. How many bumps in your city? Personalized bump seeker with mobile crowdsensing. *IEEE Transactions on Instrumentation and Measurement* 71, pp. 1–12.

Yin, H., Gao, H., Wang, B., Li, S., Li, J., 2022. Efficient trajectory compression and range query processing. *World Wide Web* 25 (3), pp. 1259–1285.

Yuan, H., Li, G., 2021. A survey of traffic prediction: from spatio-temporal data to intelligent transportation. *Data Science and Engineering* 6, pp. 63–85.

Zhang, T., Ramakrishnan, R., Livny, M., 1997. BIRCH: A new data clustering algorithm and its applications. *Data mining and knowledge discovery* 1 (2), pp. 141–182.

Zhao, Y., Li, S., Hu, S., Su, L., Yao, S., Shao, H., Wang, H., Abdelzaher, T., 2017. Greendrive: A smartphone-based intelligent speed adaptation system with real-time traffic signal prediction. In: *Proceedings of the 8th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS).* pp. 229–238.

Zheng, A., Casari, A., 2018. Feature engineering for machine learning: principles and techniques for data scientists, 1. Edition. O'Reilly Media, Inc.

Zheng, Y., Fu, H., Xie, X., Ma, W.-Y., Li, Q., 2011. Geolife gps trajectory dataset - user guide. Geolife gps trajectories 1.1 Edition.

Zhou, X., Zhang, L., 2016. Crowdsourcing functions of the living city from Twitter and Foursquare data. *Cartography and Geographic Information Science* 43 (5), pp. 393–404.

Zhu, Y., Liu, X., Wang, Y., 2013. Pervasive urban sensing with large-scale mobile probe vehicles. *International Journal of Distributed Sensor Networks* 2013.

Zimmermann, M., Kirste, T., Spiliopoulou, M., 2009. Finding stops in error-prone trajectories of moving objects with time-based clustering. vol. 53. pp. 275–286.

Zourlidou, S., Fischer, C., Sester, M., 2019. Classification of street junctions according to traffic regulators. In: *Kyriakidis, P., Hadjimitsis, D., Skarlatos, D. & Mansourian, A. (Eds.), 2019. Accepted Short Papers and Posters from the 22nd AGILE Conference on Geo–information Science. Cyprus University of Technology 17–20 June 2019, Limassol, Cyprus.*

Zourlidou, S., Golze, J., Sester, M., 2022a. [Dataset] GPS trajectory dataset of the region of Hannover, Germany. URL `https://doi.org/10.25835/9bidqxvl`

Zourlidou, S., Golze, J., Sester, M., 2022b. [Dataset] Traffic regulator ground-truth information for the Chicago trajectory dataset. URL `http://dx.doi.org/10.25835/0vifyzqi`

Zourlidou, S., Golze, J., Sester, M., 2022c. [Dataset] Traffic regulator ground-truth information of the city of Hannover, Germany. URL `https://doi.org/10.25835/cqg0x1el`

Zourlidou, S., Golze, J., Sester, M., 2022d. Traffic regulation recognition using crowd-sensed gps and map data: a hybrid approach. *AGILE: GIScience Series* 3, pp. 22.

Zourlidou, S., Sester, M., 2015a. Road regulation sensing with in-vehicle sensors. In: *A. Comber, B. Bucher and S. Ivanovic (Eds.): Proceedings of the 3rd AGILE Phd School, Champs sur Marne, France, 15–17 September 2015, published at http://ceur-ws.org.*

Zourlidou, S., Sester, M., 2015b. Towards regulation-aware navigation: a behavior-based mapping approach. In: *Accepted Short Papers from the 18th AGILE Conference on Geographic Information Science, Lisbon, Portugal.*

Zourlidou, S., Sester, M., 2019. Traffic regulator detection and identification from crowdsourced data–a systematic literature review. *ISPRS International Journal of Geo-Information* 8 (11).

Zourlidou, S., Sester, M., Hu, S., 2023. Recognition of intersection traffic regulations from crowdsourced data. *ISPRS International Journal of Geo-Information* 12 (1).

Stefania Zourlidou

Curriculum Vitae

Personal Data

Date and Place of Birth: Panorama Thessaloniki, Greece, 09.03.1981

Education

2014–2023	**P**h.D. student **Leibniz Universität Hannover, Germany** Institute of Cartography and Geoinformatics Thesis Topic: "Traffic Regulation Recognition from GPS Trajectories" Supervisor: Prof. Dr.-Ing. habil. Monika Sester
2016–2017	Promotion plus+ qualifiziert. **Graduiertenakademie, Leibniz Universität Hannover** A two-semester program that provides doctoral students with management competencies for career outside academia
2004–2005	MS.c. in Intelligent Systems **University College London (UCL), U.K.** Department of Computer Science Thesis Topic: "Machine Learning Estimation for Financial Time Series". Supervisor: Prof. Dr.-Ing. Fernando Perez-Cruz
1999–2004	**D**egree in Computer Science **University of Ioannina, Greece** Department of Computer Science Thesis Topic: "Development of a Computational System for the Estimation of Human Quadriceps Muscles Length Variation, with the help of a Gait Analysis System" Supervisor: Prof. Dr.-Ing. Chrysostomos Stylios

Professional Experience

08.2014– 26.12.2018 (full-time) 27.12.2018– (part-time)	**R**esearch Associate **Leibniz Universität Hannover, Germany** Institute of Cartography and Geoinformatics Total Maternity Leave: 29 months ≈ 2.5 years
08.2014– 07.09.2017	**E**xternal Research Associate **IAV Automotive Engineering, Gifhorn, Germany**
10.2005– 08.2013	**V**arious Teaching (4 years) and Administrative (4 years) Positions **Greek Ministry of National Education and Religious Affair, Greece**

Acknowledgements

I would like to express my deep gratitude to the two professors, *Prof. Christian Heipke* and *Prof. Jörg Müller*, who were on the dissertation committee, for their valuable recommendations and comments. Moreover, I would like to express my gratitude to the following persons:

My supervisor *Prof. Monika Sester*: I thank her for the time she spent in our one-to-one meetings, her ideas, her critical ear, for keeping an open door, and motivating me to find alternatives instead of insisting on fixing things that do not work as expected. This saved me a lot of time and I have since established it as a rule of my life. I thank her for her understanding in the numerous challenges I faced during my studies. Any opportunity related to research I asked her for, she accepted and did her best to support me. Above all I thank her for her enthusiasm and high-performance culture which indeed is infectious!

From my colleagues at ikg: special thanks to *Prof. Claus Brenner* for his comments on my presentations at the institute and his insights. Some of his hilarious comments during our institute meetings at ikg I will never forget! Many thanks to *Colin Fischer* for helping me somewhere in the middle of my studies to move forward by suggesting a very different way of dealing with a problem I was stuck on. There was a time in my research when I felt I had hit a wall and although the solution we worked on together is not included in this thesis, thanks to this collaboration I got the chance to look at the problem from a new viewpoint and arrived at the solution I was looking for. Colin, I am grateful to you for this. In addition, I thank *Jens Golze, Hao Cheng and Udo Feuerhake* for all the projects we worked on together. I learned a lot from all of you! In addition, I thank my colleague, friend and talented cook *Anna Malinovskaya* for lunch-breaks, recipes, birthday gifts and visits during my maternity leave. I also thank the ikg secretary Evelin Schramm for her friendly personality, beautiful birthday cards and support in various procedural matters. Finally, I thank my hiwi students, *Maximilian Idahl, Swarnalatha Pasam* and *Yuanhang Wu* for supporting me on various tasks in my research, and to the students I supervised on their postgraduate thesis, as I learn from them too! Finally, I deeply thank *Dr. Shaohan Hu* for providing access to a valuable dataset at a time when I was desperately looking for data to test and compare my methodology.

I am grateful to the *Equal Opportunities Office (Hochschulbüro für ChancenVielfalt)* of Leibniz University Hanover for supporting me financially during the last six months of my studies in order to complete and submit my thesis. Ms. *Elke Buchholz*, head of the office, helped me to complete the application and thanks to the funding I received, I was able to complete my dissertation. Without this support, I am not sure if I would have been able to write the dissertation, working another job and taking care of two young children. I am also grateful to *IAV GmbH* for supporting my research for three years, and to the *Deutsche Forschungsgemeinschaft (DFG)* for also supporting my work at various stages.

I cannot express in words my gratitude to my parents. They both worked very hard throughout their lives to raise and educate me and my sisters and to provide us with everything we needed to fulfill our dreams for our future. I am also grateful to my sisters for their love, encouragement and for always being there for me. I will be forever grateful for you.

Finally, I thank my husband for his support and love over the years, but most of all for creating a family together that is beyond any dream I could have ever had. Maria and Sofia, you are the brightest stars in the sky of my soul. I love you.

Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik der Leibniz Universität Hannover

(Eine vollständige Liste der Wiss. Arb. ist beim Geodätischen Institut, Nienburger Str. 1, 30167 Hannover erhältlich.)

Nr. 357	MAAS, Alina Elisabeth:	Klassifikation multitemporaler Fernerkundungsdaten unter Verwendung fehlerbehafteter topographischer Daten (Diss. 2020)
Nr. 358	NGUYEN, Uyen:	3D Pedestrian Tracking Using Neighbourhood Constraints (Diss. 2020)
Nr. 359	KIELER, Birgit:	Schema-Matching in räumlichen Datensätzen durch Zuordnung von Objektinstanzen (Diss. 2020)
Nr. 360	PAUL, Andreas:	Domänenadaption zur Klassifikation von Luftbildern (Diss. 2020)
Nr. 361	UNGER, Jakob:	Integrated Estimation of UAV Image Orientation with a Generalised Building Model (Diss. 2020)
Nr. 362	COENEN, Max:	Probabilistic Pose Estimation and 3D Reconstruction of Vehicles from Stereo Images (Diss. 2020)
Nr. 363	GARCIA FERNANDEZ, Nicolas:	Simulation Framework for Collaborative Navigation: Development - Analysis – Optimization (Diss. 2020)
Nr. 364	VOGEL, Sören:	Kalman Filtering with State Constraints Applied to Multi-sensor Systems and Georeferencing (Diss. 2020)
Nr. 365	BOSTELMANN, Jonas:	Systematische Bündelausgleichung großer photogrammetrischer Blöcke einer Zeilenkamera am Beispiel der HRSC-Daten (Diss. 2020)
Nr. 366	OMIDALIZARANDI, Mohammad:	Robust Deformation Monitoring of Bridge Structures Using MEMS Accelerometers and Image-Assisted Total Stations (Diss. 2020)
Nr. 367	ALKHATIB, Hamza:	Fortgeschrittene Methoden und Algorithmen für die computergestützte geodätische Datenanalyse (Habil. 2020)
Nr. 368	DARUGNA, Francesco:	Improving Smartphone-Based GNSS Positioning Using State Space Augmentation Techniques (Diss. 2021)
Nr. 369	CHEN, Lin:	Deep learning for feature based image matching (Diss. 2021)
Nr. 370	DBOUK, Hani:	Alternative Integrity Measures Based on Interval Analysis and Set Theory (Diss. 2021)
Nr. 371	CHENG, Hao:	Deep Learning of User Behavior in Shared Spaces (Diss. 2021)
Nr. 372	MUNDT Reinhard Walter:	Schätzung von Boden- und Gebäudewertanteilen aus Kaufpreisen bebauter Grundstücke (Diss. 2021)
Nr. 373	WANG, Xin:	Robust and Fast Global Image Orientation (Diss. 2021)
Nr. 374	REN, Le:	GPS-based Precise Absolute and Relative Kinematic Orbit Determination of Swarm Satellites under Challenging Ionospheric Conditions (Diss. 2021)
Nr. 375	XU, Wei:	Automatic Calibration of Finite Element Analysis Based on Geometric Boundary Models from Terrestrial Laser Scanning (Diss. 2021)
Nr. 376	FENG, Yu:	Extraction of Flood and Precipitation Observations from opportunistic Volunteered Geographic Information (Diss. 2021)
Nr. 377	YANG, Chun:	A hierarchical deep learning framework for the verification of geospatial databases (Diss. 2021)
Nr. 378	MEHLTRETTER, Max:	Uncertainty Estimation for Dense Stereo Matching using Bayesian Deep Learning (Diss. 2021)
Nr. 379	KAZIMI, Bashir:	Self Supervised Learning for Detection of Archaeological Monuments in LiDAR Data (Diss. 2021)
Nr. 380	PETERS, Torben:	Learning Multi-View 2D to 3D Label Transfer for Semi-Supervised Semantic Segmentation of Point Clouds (Diss. 2022)
Nr. 381	WASSINK, Martin:	Kommunal- und Regionalentwicklung durch Kooperation und Teilung von Verantwortung in ländlichen Räumen - eine multiperspektivische Untersuchung an Beispielen aus dem Raum Steinwald/Fichtelgebirge (Diss. 2022)
Nr. 382	GOLDSCHMIDT, Jürgen:	Die Berücksichtigung künftiger Entwicklungen bei der Verkehrswertermittlung (Diss 2022)
Nr. 383	KRUSE, Christian:	Impact maps from bomb craters detected in aerial wartime images using marked point processes (Diss. 2023)
Nr. 384	ZOURLIDOU, Stefania:	Traffic Regulation Recognition from GPS Data (Diss. 2023)