Methods for Event Detection and Classification for Radar-Based Bridge Monitoring

Matthias Oliver Arnold

Doctoral Thesis
Karlsruhe, 2024

Methods for Event Detection and Classification for Radar-Based Bridge Monitoring

Zur Erlangung des akademischen Grades eines

DOKTORS DER INGENIEURWISSENSCHAFTEN (Dr.-Ing.)

von der KIT Fakultät für
Bauingenieur-, Geo- und Umweltwissenschaften
des Karlsruher Instituts für Technologie (KIT)

genehmigte

DISSERTATION

von

Matthias Oliver Arnold
geboren in Tettnang

Tag der mündlichen Prüfung:	28.06.2024
Hauptreferentin:	PD Dr. rer. nat. Sina Keller Institut für Photogrammetrie und Fernerkundung (IPF) Karlsruher Institut für Technologie (KIT)
1. Korreferentin:	Prof. Dr. Corinna Harmening Geodätisches Institut (GIK) Karlsruher Institut für Technologie (KIT)
2. Korreferent:	Prof. Dr.-Ing. Stefan Hinz Institut für Photogrammetrie und Fernerkundung (IPF) Karlsruher Institut für Technologie (KIT)

Karlsruhe, 2024

Matthias Oliver Arnold

Methods for Event Detection and Classification for Radar-Based Bridge Monitoring

Doctoral Thesis

Date of examination: 28 June 2024

Reviewers: PD Dr. rer. nat. Sina Keller, Prof. Dr. Corinna Harmening and Prof. Dr.-Ing. Stefan Hinz

Karlsruhe Institute of Technology

Institute of Photogrammetry and Remote Sensing (IPF)

Department of Civil Engineering, Geo and Environmental Studies

Kaiserstr. 12

76131 Karlsruhe

Abstract

With the continuous growth in heavy goods traffic, bridges are exposed to increasing loads. This trend impacts especially older bridges originally designed with lower requirements. The extent to which they are exposed to heavy-duty traffic, however, is site-specific. Thus, only some bridges need expensive reinforcements. Localized data about traffic can, therefore, give valuable insights concerning the bridge's durability. In order to acquire traffic information, Bridge Weigh-in-Motion (BWIM) systems are oftentimes used. These systems usually exploit permanently attached sensors, leading to costly installation and maintenance. These costs scale with the number of monitored bridges. Ground-Based Radar (GBR) has recently emerged as an alternative, flexible solution that remotely measures bridge displacement. So far, GBR systems have not been used as sensors for BWIM. This thesis elaborates on how suitable GBRs are for automatically extracting traffic information. Two bridges have been monitored throughout this work to collect the respective data. An Unmanned Aerial Vehicle has recorded the upper side of the bridge to acquire reference data.

In the first part of the thesis, we investigate data-driven methods for the automatic detection of vehicles crossing a bridge, i.e., events. To this end, the displacement time series is split in windows of $0.5\,\mathrm{s}$ and the dataset is randomly shuffled. Both shallow learning and Deep Learning (DL) approaches are used to determine the presence of a vehicle for each window. DL, in the shape of a Convolutional Neural Network (CNN), has achieved very good results on individual windows. In the final analysis of the first part, we evaluate the CNN's ability to segment events from start to finish in a real-world use case with ordered windows of a measurement campaign. A threshold-based approach is used as a baseline. Overall, the CNN can succesfully extract entire events from bridge displacement signals.

The second part of this thesis focuses on extracting traffic information from detected events. First, we aim to classify events according to the vehicles involved with the ulterior motive of acquiring single events, i.e. events caused by only one vehicle. Again, we apply data-driven shallow learning. Due to the small and imbalanced dataset, we investigate the benefits of data augmentation. Data augmentation's effect is analyzed thoroughly using Explainable Artificial Intelligence methods. Ultimately, we show that single events can be identified correctly.
In the next step, we extract different vehicle parameters from single events: vehicle

type, lane, locus, speed, and the presence of trailers. For vehicles without trailers, the number of axles and the spacing between axles is estimated. We show that, in principle, data-driven methods are suitable to determine these vehicle parameters. Furthermore, we demonstrate at one bridge that axles can cause high-frequency vibrations recorded by the GBR.

In the third and final part, we combine the best models of all previous studies into a single pipeline. Simulating a real-world use case, we show that this pipeline can process raw bridge displacement data and extract vehicle parameters successfully.

Zusammenfassung

Durch den kontinuierlichen Anstieg an Schwerlastverkehr, sind Brücken immer höheren Belastungen ausgesetzt. Dieser Trend betrifft vor allem ältere Brücken, die ursprünglich mit geringeren Anforderungen geplant wurden. Das Ausmaß der Belastung ist jedoch ortsabhängig. Demenstprechend benötigen nicht alle Brücken teure Verstärkungen. Daten über den lokalen Verkehr können daher wertvolle Erkenntnisse über die Standhaftigkeit von Brücken liefern. Sogenannte Bridge Weigh-in-Motion (BWIM)-Systeme werden häufig zur Sammlung von Verkehrsdaten eingesetzt. Diese Systeme nutzen für gewöhnlich Sensoren, die fest angebracht werden, was zu hohen Installations- und Instandhaltungskosten führt. Diese Kosten erhöhen sich mit der Anzahl an überwachten Brücken. Bodenbasiertes interferometrisches Radar (engl.: Ground-Based Radar - GBR) hat sich in letzter Zeit als alternative, flexible Lösung zur Messung von Brückenverschiebungen aus der Ferne etabliert. Bisher wurden GBR-Systeme nicht als Sensoren für BWIM eingesetzt. Diese Arbeit untersucht die Tauglichkeit von GBR-Systemen zur automatisierten Bestimmung von Verkehrsinformationen. Im Verlauf der Arbeit wurden Daten von zwei Brücken gesammelt. Um Referenzdaten zu erhalten, hat eine Drohne (engl.: Unmanned Aerial Vehicle - UAV) die Brückenoberseite videoüberwacht.

Im ersten Teil der Arbeit untersuchen wir daten-getriebene Methoden zur automatischen Erkennung von Brückenüberfahrten bzw. Events. Dazu wird die Verschiebungszeitreihe in $0.5\,\mathrm{s}$ lange Fenster unterteilt und der entstandene Datensatz zufällig gemischt. Mithilfe von oberflächlichem Lernen (engl.: Shallow Learning) und tiefem Lernen (engl.: Deep Learning - DL) wird für jedes Fenster bestimmt, ob sich ein Fahrzeug auf der Brücke befindet. DL hat in Form eines faltungs-basierten neuronalen Netzwerks (engl.: Convolutional Neural Network - CNN) bei der Klassifikation einzelner Fenster sehr gute Ergebnisse erzielt. Abschließend wird im ersten Teil anhand einer realen Messkampagne ausgewertet, wie gut sich das CNN zur Segmentierung von Events von Anfang bis Ende eignet. Ein schwellwert-basierter Ansatz dient als Referenz. Insgesamt kann das CNN Events erfolgreich aus Brückenverschiebungsdaten extrahieren.

Der zweite Teil dieser Arbeit konzentriert sich darauf, Verkehrsdaten aus erkannten Events zu extrahieren. Zunächst soll die Anzahl an Fahrzeugen in einem Event

bestimmt werden, wobei wir letztlich darauf abzielen, sogenannte Einzelevents, d.h. Events mit nur einem Fahrzeug, zu erkennen. Dazu setzen wir erneut auf Shallow Learning. Da unser Datensatz klein und unausgeglichen ist, untersuchen wir zusätzlich Datenaugmentation. Der Effekt der Augmentation wird mithilfe von Methoden der erklärbaren künstlichen Intelligenz sorgfältig analysiert. Letztlich zeigen wir, dass Einzelevents korrekt erkannt werden können.
Im nächsten Schritt werden verschiedene Parameter aus Einzelevents extrahiert: Fahrzeugtyp, Fahrspur, Lokus, Geschwindigkeit sowie das Vorhandensein von Anhängern. Bei Fahrzeugen ohne Anhänger werden zusätzlich noch die Anzahl an Achsen und die Abstände zwischen Achsen bestimmt. Wir zeigen, dass diese Fahrzeugparameter mithilfe von daten-getriebenen Verfahren grundsätzlich bestimmt werden können. Außerdem demonstrieren wir an einer Brücke, dass Fahrzeugachsen hoch-frequente Schwingungen auslösen können, welche vom GBR aufgezeichnet werden.

Im dritten und letzten Teil kombinieren wir die besten Modelle aller vorangegangenen Studien in einer Pipeline. Durch die Simulation eines realen Anwendungsfalls zeigen wir, dass diese Pipeline erfolgreich rohe Brückenverschiebungsdaten verarbeiten und Fahrzeugparameter extrahieren kann.

Acknowledgement

First of all, I want to thank my advisor Sina Keller for giving me the opportunity to start a PhD and also for seeing me all the way through. You always had my back and no matter what kind of absurd challenges I enountered, you always came up with splendid suggestions. The same goes for Stefan Hinz, who always provided valuable input due to his quick comprehension.

Secondly, I want to thank the great team at ci-tec where I worked during my PhD. Mareike, Raoul, André, Marc (and unfortunately not anymore: Laura and Caramel) you make going to work quite enjoyable and I am really happy to work together with all of you. Thank you all for your input, for enhancing my ordinary writing and for listening to my overeager ideas. This work became what it is now only due to your help.

Furthermore, I want to thank Chrissy, Felix and Pascal for helping me out with information regarding cars. Felix and Pascal, I am still impressed by how well you identified cars from low-quality images. Anja, thank you for reading and correcting all of my writing despite being on vacation and moving into a new home. Knowing that you would proof-read everything in the end really eased my mind. My thanks also go to Andreas and Chris, who somehow turned waiting under a bridge for measurements to end into a fun experience. Big thanks to my family for being there for me and providing me with tea and chocolate.

Last but not least, I want to thank my wonderful girlfriend, of course. I am sorry I spent so much my time in front of a screen instead of with you. Yet, you always supported and motivated me. I hope I can return the favour eventually.

I apologize for everyone I did not mention here but in my defense, I am really, really tired. Thank you though.

Contents

Introduction

1

If watching a bridge is much more exciting than crossing that bridge, then you can be sure that it is a very beautiful bridge!

— **Mehmet Murat Ildan**
(Author and playwright)

1.1 Motivation

On 14th August 2018 the Morandi bridge in Genoa, Italy, partially collapsed leading to the death of 43 people [104]. This tragic event has triggered a fierce debate about the condition of bridges across borders and consequently in Germany as well. In Germany, the discussion swiftly changed its focused on domestic bridge conditions. Although this is undoubtedly an important discussion, the focus on the condition neglects another important factor: the impact of traffic. Bridges are designed with certain specifications concerning their load bearing capacity. Some of these specifications make assumptions about the traffic to which the bridge is exposed [97]. As heavy goods traffic becomes more excessive and more frequent, these load models need to be adapted. They are drawn from traffic measurements at various sites to cover a wide variety [46]. Still, load models cannot paint a complete picture of reality. Depending on the location of a bridge, traffic conditions might differ tremendously. This can lead to an underestimation of load exposure which jeopardizes the bridge functionality. On the other hand, traffic might be overestimated, leading to unnecessary and costly reinforcements. Finally, older bridges might have been designed with load models unfit for today's conditions. Therefore, bridge-specific traffic information would be useful to implement adequate maintenance steps.

For this purpose Bridge Weigh-in-Motion (BWIM) systems have been developed [109]. They usually consist of conventional sensors such as acceleration or strain sensors, which measure the bridges' dynamic response to vehicle crossings [106, 74]. From these signals BWIM systems ultimately try to estimate the vehicle load. Usually other traffic information such as the vehicle speed or the number of axles need to be known in order to determine the load. Thus, several parameters need to be extracted from the bridge response. A disadvantage of conventional sensors is that they have to be permanently attached to the bridge. This might inflict damage to the bridge and requires access to the infrastructure. Additionally, system maintenance can become very circumstantial. Despite, non-invasive solutions have barely been studied [115]. The development of a new monitoring method has been part of the project ZEBBRA[1] in the context of which some of the thesis studies' were written. It deploys Ground-Based Radar (GBR) to remotely measure the bridge deflection.

GBR has emerged several years ago as a solution for bridge monitoring [120]. Since then, multiple studies have researched its possibilities [14, 31, 62, 101, 102]. They often focus on extracting structural properties such as the eigenfrequency or the eigenmodes. So far, however, no survey regarding the potential of GBRs in the context of BWIM have been conducted. Despite the fact that the vehicle load can be easily extracted from the bridge displacement [115], provided detailed information about the vehicle, for instance the axle spacing, is available. One solution would be to monitor traffic with a camera [115] which increases the systems complexity. Ideally, all information is determined solely from GBR displacement data. However, acquiring the necessary vehicle parameters is not a trivial task. Machine Learning (ML) offers tools to learn complex relations between input and target values [140]. We will hence assess the potential of ML in the context of GBR-based BWIM.

1.2 Main Objective and Research Goals

The overall objective of this thesis is to develop a novel data-driven method for vehicle information extraction from GBR bridge displacement data. This is achieved upon the data collected from two bridges in Germany using data-driven ML. In order to acquire reference data, an Unmanned Aerial Vehicle (UAV) has been deployed to monitor the traffic during the GBR measurements. In general, two main goals have been defined:

[1]Eventbasierte Zustandserfassung und -bewertung von Brücken basierend auf Radardaten und intelligenten Algorithmen – english: Event-based observation and assessment of the state of bridge infrastructure based on radar data combined with intelligent algorithms. Funded by the German Federal Ministry of Education and Research [7].

Research Goal I

> The first goal is to detect bridge crossings, or events, in the GBR displacement time series data.

Research Goal II

> The second goal consists of the classification of detected events. The classification is composed of (1) determining the number of vehicles in an event, and (2) extracting vehicle parameters if only one single vehicle is crossing the bridge.

To conquer these objectives, we[2] mainly rely on ML. Especially for the second goal, several challenges are faced. For one, we can only draw on a small dataset. Secondly, the dataset is heavily imbalanced and biased depending on the task. Finally, bridge crossings are of different length. Yet, variable-length input is still often disregarded in ML, highly limiting our pool of available methods. Despite these challenges, we will show that both objectives are accomplishable. In the end, we have developed a data-driven pipeline which can process raw GBR displacement data and outputs detailed traffic information.

1.3 Thesis Outline and Contributions

In the following, the structure of this thesis is outlined. Its overall structure is visualized in Figure 1.1. Chapter 2 illustrates the GBR measurement principle and introduces to the two monitored bridges by means of example data. Afterwards, the fundamentals relevant for this thesis are explained in Chapter 3.

Part I and hence Chapter 4 deal with research goal I: the detection of events in displacement signals. We try to achieve real-time capability, meaning that vehicles are detected while they still are on top of the bridge. To this end, we classify small segments depending on whether a vehicle is currently crossing the bridge or not. First, shallow learners are investigated, trained on hand-crafted features. Second, Deep Learning (DL) is used, this time passing in the time series data directly. Finally, we evaluate how accurate events can be segmented from start to end with DL. A simple threshold-based method serves as a baseline.

[2]In order to stay consistent with quoted studies and for the sake of readability, I will use "we" throughout this work.

Part II consists of two chapters. The first one, Chapter 5, focuses on the classification of events according to the number of vehicles involved. Many BWIM methods require only one single vehicle on the bridge. For that reason, we first investigate if bridge crossing events can be distinguished in events with only one (single event) or several vehicles (multi events) using shallow learning (see Section 5.4). Since the dataset is small, imbalanced and skewed, we apply data augmentation as a countermeasure and thouroughly analyze its effect. Furthermore, we study transferability across different bridges. In pursuance of a more refined breakdown, Section 5.5 deals with estimating the exact number of vehicles contributing to an event. First of all, events are simply classified according to their vehicle count. Then, the exact amount of cars and trucks is tried to be recovered using regression. Methodologically, we follow the previous chapter to maintain comparability.
Using only single events, Chapter 6 concerns itself with extracting vehicle parameters as the second chapter of Part II. We try to determine vehicle type, presence of a trailer, lane, locus, speed, axle count, and axle spacing. Again, we utilize shallow learning. This time, however, no data augmentation is applied. Moreover, we focus on one bridge. Apart from ML, we investigate the potential of signal processing for axle detection. Digitals filters and Continuous Wavelet Transform (CWT) reveal high-frequency components in the displacement signal correlating with axle-presences. From this, even the axle spacing can be estimated to some extent.

Finally, Part III consolidates our findings of the previous parts. In Chapter 7, we merge the best performing models into a single pipeline called MONITOR (MethOds for data-driveN detection and classIficaTion of vehicles in nOn-invasive Radar data). In order to simulate a real-world usecase, all models are retrained on the first measurement campaigns and tested on the last one. Each step is evaluated and compared to the ideal scenario of perfectly segmented data. In the end, four illustrative examples are given.
Chapter 8 finishes this thesis with a summary, concluding the main objective and giving a possible outlook.

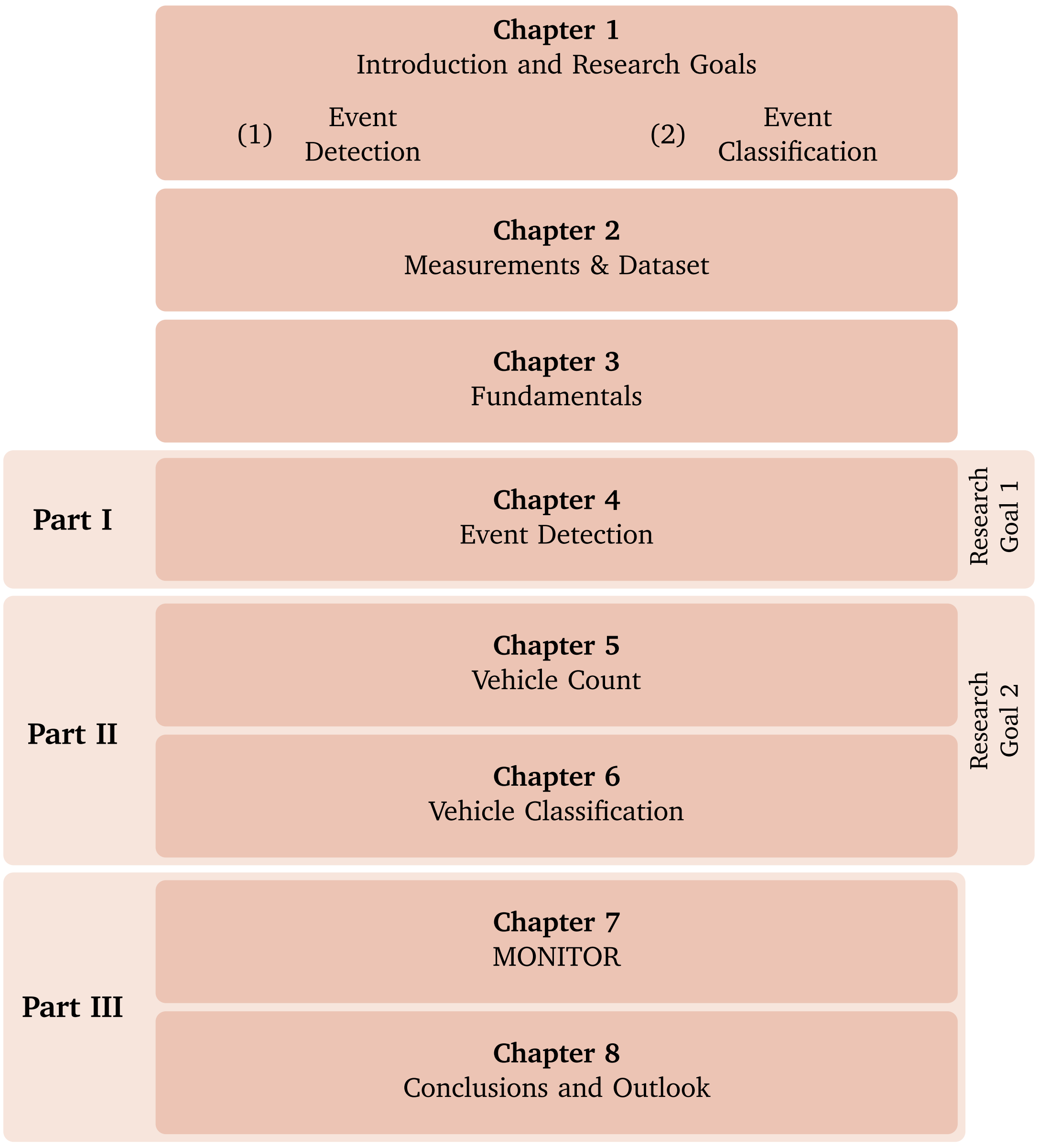


Fig. 1.1.: Structure of this thesis.

Measurements & Data 2

The distance is nothing when one has a motive.

— **Jane Austen**
(Author of Pride and Prejudice)

This chapter includes material from the following works:

M. Arnold and S. Keller. "Detection and Classification of Bridge Crossing Events With Gound-Based Interferometric Radar Data and Machine Learning Approaches". en. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* V-1-2020 (Aug. 2020), pp. 109–116

It is cited as [5] and marked with a orange line.

In this chapter, the data foundation of this work is described. This foundation includes, firstly, the GBR measurement principle and setup (see Section 2.1). Second, the relevant bridges are introduced and their respective mesurement campaigns are described in Section 2.2. The campaigns later constitute the datasets on which we develop our methods.

2.1 Radar Monitoring

A majority of bridge monitoring systems exploit invasive sensors such as acceleration or strain-gauge sensors [118]. As mentioned in Section 1.2, they have several disadvantages like a permanent attachment to one single bridge or costly maintenance. Conversely, GBR are swiftly and easily to set up and flexible in their place of usage. In this thesis, we deploy the Image by Interferometric Survey (IBIS) GBR developed by IDS GeoRadar [51]. For measurements it harnesses frequency modulation. The GBR sends out electromagnetic waves, increasing their frequency stepwise from the minimum of its bandwidth B to the maximum. The IBIS has a bandwidth of 200 MHz. Surface structures of the monitored bridge reflect these signals back to the GBR. Received waves are mixed with their corresponding originally transmitted waves to determine the beat frequency f_B. f_B is proportional to time of travel of the wave and thus the distance between the GBR and the reflecting surface. For each frequency step along beat f_B is determined. Transforming the thereby acquired spectrum to the time domain, delivers distance measures for multiple measurement points along the GBRs line of sight (LOS). The range resolution Δr is consequently dependent on the bandwidth B:

$$\Delta r = \frac{c}{2 \cdot B} = 0.75\, m, \tag{2.1}$$

where c is the speed of light [102, p. 8]. Hence, the IBIS measures the displacement in LOS for every $0.75\,\mathrm{m}$, also refered to as range bin. Its sampling frequency f_A ranges up to $200\,\mathrm{Hz}$.

To acquire the displacement, interferometry is utilized. Recieved waves are mixed with a phase-shifted signal to acquire phase information of the reflected waves. This information contains the phase difference $\Delta\phi$ between two consecutive measurements. With $\Delta\phi$ the LOS displacement ΔR results to

$$\Delta R = -\frac{\lambda}{4\pi} \cdot \Delta\phi, \tag{2.2}$$

where λ is the central wavelength of the GBR [102, p. 10]. As Equation (2.2) shows, all displacement measurement are relative and not absolute.
The GBR cannot distinguish objects within a range bin. Thus, to avoid ambiguities concerning the exact measurement point, it is necessary to have only one dominating backscatterer in each range bin. To this end, many studies attach reflectors to the monitored bridge although it might inflict structural damage [31, 100]. Besides, reflectors increase the signal-to-noise ratio (SNR).

Geometric considerations illustrated in Figure 2.1 allow a calculation of the vertical displacement Δz according to

$$\Delta z = \frac{R}{h} \cdot \Delta R, \tag{2.3}$$

where h represents the height difference between GBR and bridge, and R expresses the LOS distance between GBR and the measurement point [102, p. 10]. Figure 2.1 also includes an example displacement time series caused by a vehicle crossing a bridge. As per Equation (2.3), the displacement projection is inversely proportional to the height difference between GBR and bridge. Thus the lower the bridge, the bigger is the projection error. In other words: A bigger height of the bridge leads to a better accuracy. On the other hand, a bigger distance to the target reduces the received signal strength, i.e. the SNR. Dei et al. [33] show that a conversion

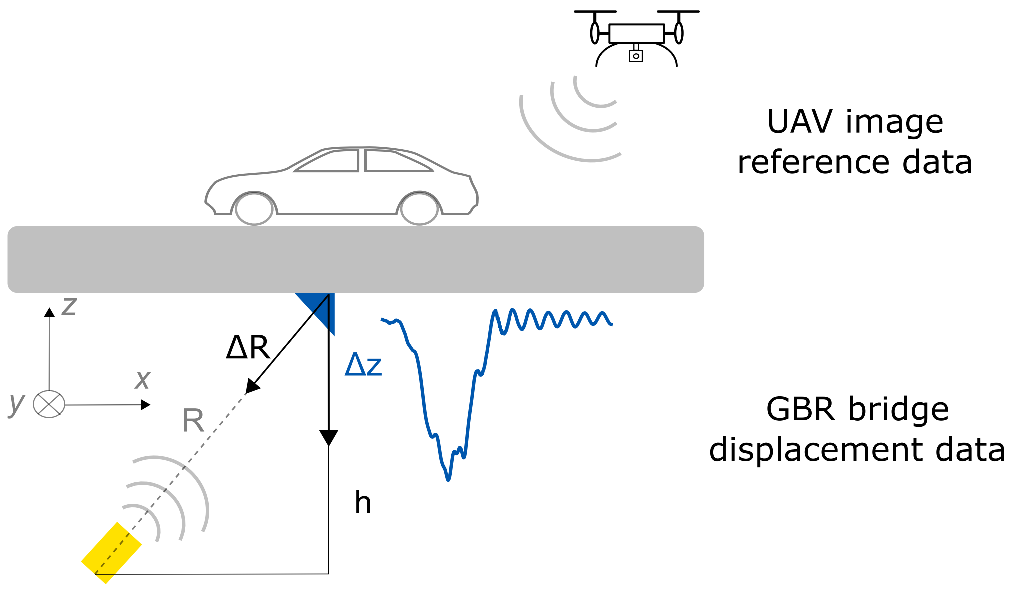


Fig. 2.1.: Schema of the measurement setup. The GBR monitors the bridge from below, measuring the displacement. An example time series is included in blue. From above an UAV records the traffic causing the displacement. Adapted from [9].

from phase difference to vertical displacement can lead to incorrect results. The reason for this is the projection of the displacement on a single component, even though bridge deflection has three degrees of freedom. Consequentially, Miccinesi et al. [99] propose a multiple-input and multiple-output (MIMO) radar to measure two components at the same time. Several studies even introduce MIMO radars for surveillance of all three degrees of freedoms. On the other hand, Michel and Keller [100] apply two single-input, single-output radars for the extraction of two displacement components. For this purpose, however, they do neglect the remaining third component. In this work, we rely mainly on the vertical displacement.

Figure 2.2 shows a several hours long displacement measurement of reflector 1 of Bridge A (see Section 2.2). The large peaks are crossings by heavy duty vehicles. Smaller peaks are caused by cars. As can be seen, GBR measured displacement signals are characterised by a drift in a non-linear manner over time. Similar to all electronic devices, the GBR is influenced by environmental parameters of the measurement surroundings such as temperature, relative humidity and air pressure. However, these parameters affect the GBR device on the one hand and the GBR signal during the transmission on the other hand. Furthermore, the bridge is also affected by the dynamics of the environmental parameters. For example, the bridge expands under rising temperature which leads to a slow horizontal movement of the reflectors. All these aspects can contribute to the long-term drift, making it difficult to model their effect.

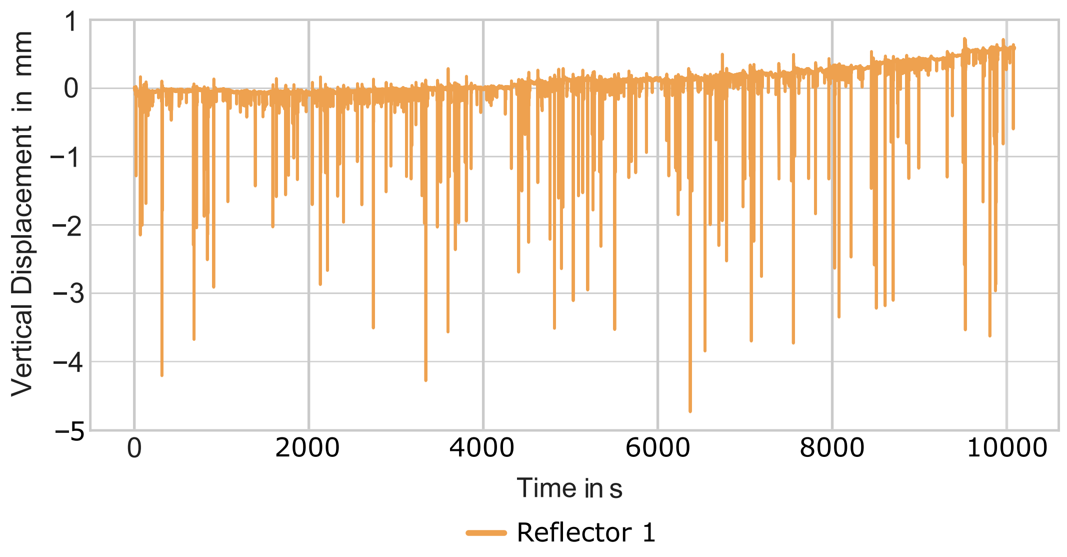


Fig. 2.2.: The vertical displacement of reflector 1 as a time series over about 10 000 s. The occurring peaks in the time series correspond to different events. The time-series data has been low-pass filtered. Adapted from [5].

2.2 Bridges

In this section, a brief introduction to the monitored bridges will be given. For each bridge, we furthermore provide an overview of the respective dataset. As illustrated by Figure 2.1, an UAV has been deployed during measurements to record traffic on top of the bridge. Two GBRs monitor the lower side concurrently. By combining both GBR and UAV data, a database has been constructed containing information about vehicles and the bridge displacement they cause. The UAV data serves as reference in this work. This section will only give an overview of the datasets. Relevant details are discussed in the chapters using them.

Both bridges lie near Coburg, in the southeast of Germany. One is close to a town called Dietersdorf, the other one close to Mödlitz. From now on, **Bridge A** refers to the bridge near Dietersdorf and **Bridge B** to the one near Mödlitz. Table 2.1 provides an overview of relevant details including the natural frequency extracted from measurements.

Tab. 2.1.: Overview of selected bridge details. Adopted from [5].

Name	Type	Fields	Length	Width	Height	Natural Frequency
Bridge A	Beam/plate mixing system	2	57.0 m	13.6 m	5.9 m	3.6 Hz
Bridge B	Plate girder bridge/ Girder Grid Bridge	1	26.36 m	11.69 m	9.15 m	3.7 Hz

2.2.1 Bridge A

The bridge near Dietersdorf consists of two $28.5\,\mathrm{m}$ long fields whereof one was equipped with five reflectors. Below this field there is no street but a flooding channel. The federal highway B303 leads over the bridge with two lanes, one for each driving direction. The speed limit is $100\,\mathrm{km\,h^{-1}}$. Both fields are connected via the asphalt and a concrete plate. This weak connection becomes evident with heavy vehicles when a lifting of the field is visible in the GBR signal while the vehicle is on the other field. The top and middle image of Figure 2.3 present the bridge from two different perspectives. The bridge is approximately $9.15\,\mathrm{m}$ high. Since Bridge A has a rectangular shape it was necessary to offset the reflectors to avoid ambiguities within a range bin. Five reflectors as well as three temperature sensors have been installed. Inbetween reflectors a distance of approximately 2 meters has been established to assure that they lie in different range bin cells. All reflectors are at the same altitude and visible from both GBR positions. The GBRs have been setup such that their LOS' are orthogonal. Thus, they measure the vertical displacement as well as primarily either the x- or y-component.
The bottom figure of Figure 2.3 depicts a vertical displacement time series. It is caused by the truck displayed in the image above and measured by the GBR beneath the bridge. Comparing the relative maximum displacement of all reflectors, the driving side can be determined. Also, reflector 5 is influenced be short-term drifts. A higher noise is to be expected, since it is the furthest away from the GBR, but there also seem to be some other factors. One possible explanation is that adiditonal unintended reflections occur within this range bin. Finally, the bridge bending curvature is superimposed by the bridge oscillation of approximately $3.6\,\mathrm{Hz}$. More examples of vehicles and the displacement signals can be admired in Appendix A.1.

Five UAV-supported measurement campaigns have been conducted at Bridge A. Table 2.2 lists their dates as well as the number of vehicles recorded during the UAV flight time and the covered air temperature range. Altogether, 1110 vehicles have been recorded at Bridge A.

Tab. 2.2.: Overview of measurement campaigns conducted at Bridge A.

Date	Number of vehicles	Temperature range
23.10.2019	202	$11.2\,°\mathrm{C}$ to $12.4\,°\mathrm{C}$
24.10.2019	79	$9.0\,°\mathrm{C}$ to $9.1\,°\mathrm{C}$
27.02.2020	438	$3.8\,°\mathrm{C}$ to $5.0\,°\mathrm{C}$
29.07.2020	338	$18.1\,°\mathrm{C}$ to $19.4\,°\mathrm{C}$
09.06.2021	53	$21.6\,°\mathrm{C}$ to $21.9\,°\mathrm{C}$

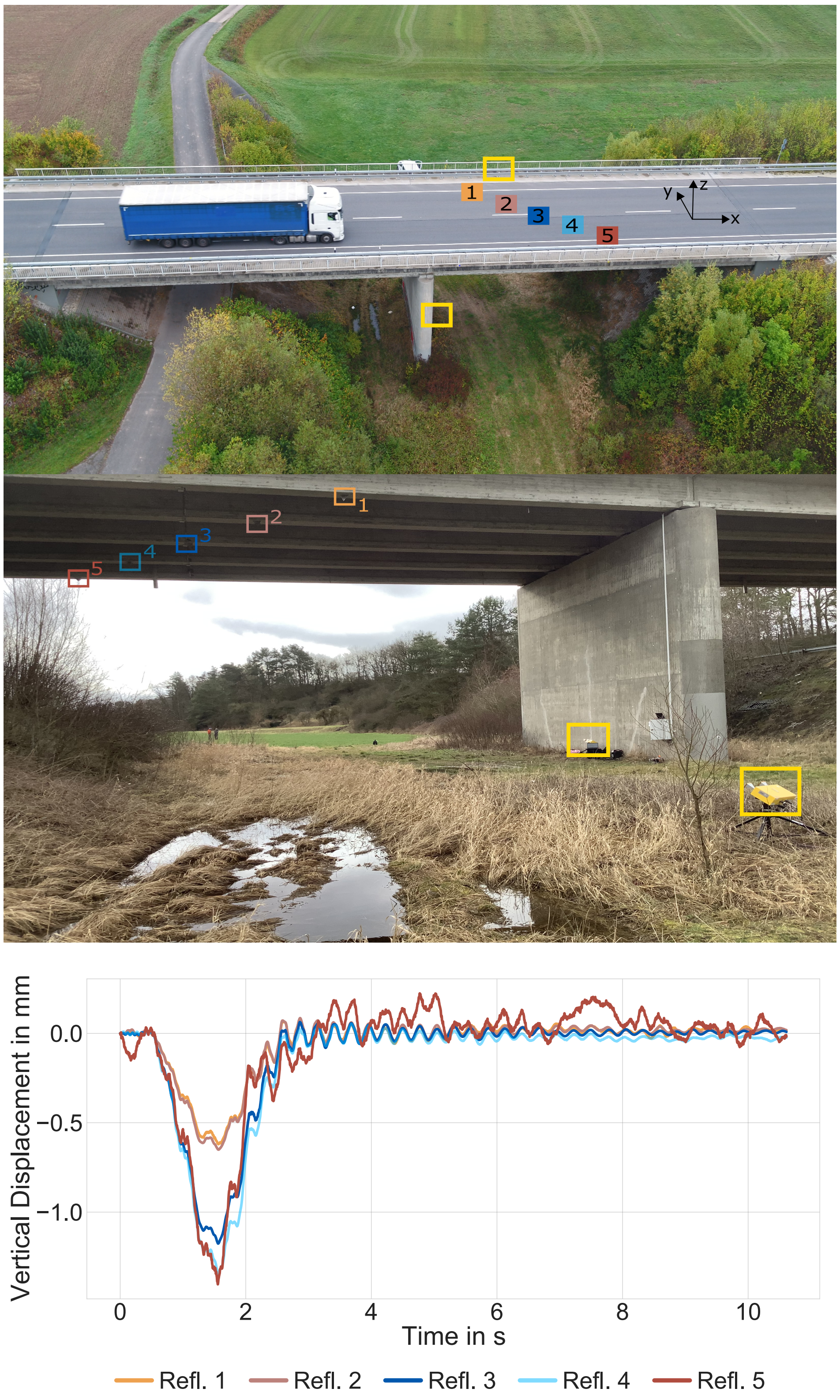


Fig. 2.3.: Top: View of Bridge A recorded by an UAV, with marked positions of both GBR and all five reflectors. Middle: Lower deck of the measurement setup of Bridge A. The GBR and five reflectors are highlighted. Bottom: Displacement time series recorded by the GBR measuring along the x-axis for the truck shown above.

2.2.2 Bridge B

The bridge near Mödlitz consists of one field and represents a common type of freeway bridge. Above the bridge also runs the federal highway B303 with a speed limit of $100\,\mathrm{km\,h^{-1}}$. Below it runs road K29. It has an approximate height of $5.9\,\mathrm{m}$ and thus a very small height difference h between bridge and GBR, leading to a larger projection error compared to Bridge A. Figure 2.4 shows different perspectives on the bridge as well as the displacement time series of one example event. Both GBR are highlighted by yellow boxes. Three reflectors have been attached to the lower side of the bridge: two beneath the beams, one in between. The latter has a higher vertical position. Only the GBR measuring parallel to the road sees the central reflector 2 highlighted in red in Figure 2.4. Since the bridge surface is not rectangular but has the shape of a parallelogram, it was possible to place all three reflectors in the center of the bridge but still have them in different range bins. Thus, they can be distinguished by the GBR beneath the bridge. Three temperature sensors measure the air, concrete and asphalt temperature. The bottom figure of Figure 2.4 depicts the time series for all three reflectors caused by the truck included in the image above. Apart from the bending of the bridge, a slight oscillation with the dominating natural frequency of around $3.7\,\mathrm{Hz}$ can be seen for reflector 1 and 2. It superimposes the bending and continues to decay after the truck leaves the bridge. Finally, the used lane cannot be deduced from the relative maximum displacement of reflector 1 compared to reflector 2.
Reflector 3 is special in two aspects: Firstly, the signal is very noisy even though refector 3 is close to the other two reflectors. Vehicles, especially trucks, which pass below the bridge interfere with the signal. This can lead to broadband disturbances, phase jumps or even a full abortion of the measurement by the GBR software. In order to mitigate the effect, the tilt of the GBR radar below the bridge was increased [102]. As a consequence the SNR of all reflectors is not optimal, especially for reflector 3. Moreover, disturbances and phase jumps still occur due to traffic. Secondly, the signal shape of reflector 3 differs tremendously from the other two. Reason for that might be the lane of the truck and the fact that the GBR measures more than one displacement component Michel and Keller [100]. A high displacement in the x-component (here the vertical driving direction - see Figure 2.4) due to heavy traffic might lead to the depicted waveform. This behaviour is not the case for all vehicles as can been seen in the additional examples included in Appendix A.2.

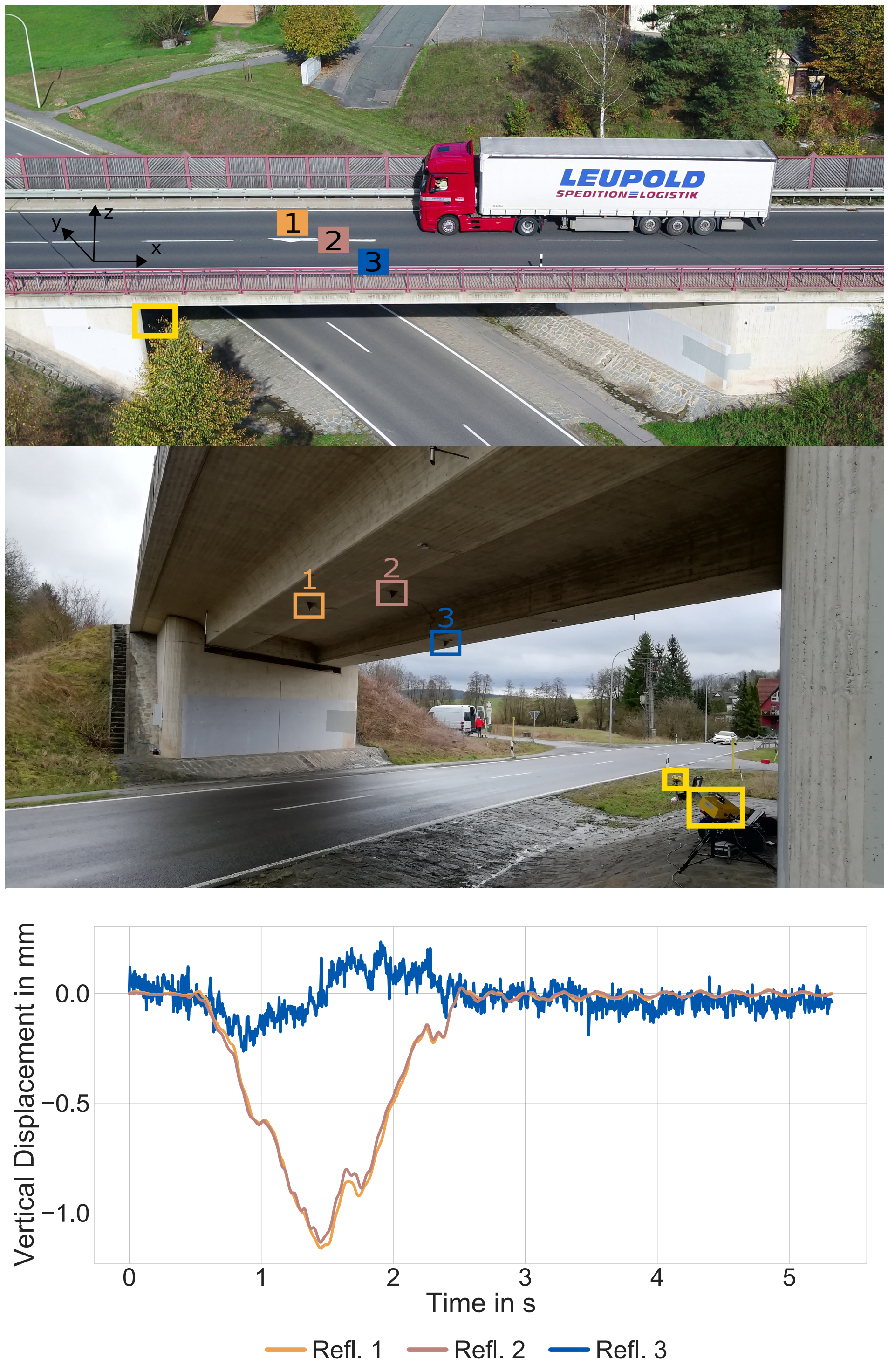


Fig. 2.4.: Top: View of Bridge B recorded by an UAV, with marked positions of one GBR and all three reflectors. Middle: Lower deck of the measurement setup of Bridge B. The GBR and three reflectors are highlighted. Bottom: Displacement time series of the GBR beneath the bridge caused by the truck shown above.

Tab. 2.3.: Overview of measurement campaigns conducted at Bridge B.

Date	Number of vehicles	Temperature range
22.10.2019	339	1.5 °C to 13.8 °C
25.02.2020	152	7.2 °C to 8.0 °C

Table 2.3 gives an overview of measurements campaigns at Bridge B included in this work. As mentioned before, a traffic runs below Bridge B causing disturbances and phase jumps in the displacement signal. In this work, we only regard disturbance-free events. In sum, 491 undisturbed vehicles have been recorded.

3 Fundamentals

Don't forget to smile in any situation. As long as you are alive, there will be better things later, and there will be many.

— **Eiichiro Oda**
(Mangaka)

This chapter includes material from

Matthias Arnold and Sina Keller. "Machine Learning and Signal Processing for Bridge Traffic Classification with Radar Displacement Time-Series Data". en. In: *Infrastructures* 9.3 (Mar. 2024). Number: 3 Publisher: Multidisciplinary Digital Publishing Institute, p. 37

It is cited as [8] and marked with a green line.

In this chapter, we will explain the necessary fundamentals of this thesis. First, we will describe various relevant signal processing methods in Section 3.1. This includes multiple time series features used in this study, such as digital filters and CWT. Note that we apply the listed features in the context of ML and do not regard them as statistical properties. This distinction should be made, as the primary goal of feature engineering in ML is to find valuable representations of the input data independent of measurements, while statistics requires incorporating knowledge about the system [20]. Afterwards, we will give an overview of ML methods in Section 3.2. Therein, we describe data preprocessing, shallow and deep learning methods, and different evaluation metrics. Moreover, we detail methods of Explainable Artificial Intelligence (AI).

3.1 Signal Processing

Many sensors, such as GBRs, measure analog time series signals which are then digitalized and processed. We define a discrete univariate time series $\underline{x}$ of length N as

$$\underline{x} = (x_0, x_1, ..., x_{N-1}), \tag{3.1}$$

where x_i represents a measurement at a time step i. All following equations are adapted to the nomenclature of our time series definition.

For a multivariate time series $\underline{x}_{MV}$, k values are measured concurrently in each time step i. $\underline{x}_{MV}$ is therefore defined as a vector of vectors:

$$\underline{x}_{MV} = \left(\begin{bmatrix} x_0^0 \\ x_0^1 \\ \vdots \\ x_0^{k-1} \end{bmatrix}, \ldots, \begin{bmatrix} x_{N-1}^0 \\ x_{N-1}^1 \\ \vdots \\ x_{N-1}^{k-1} \end{bmatrix} \right). \tag{3.2}$$

3.1.1 Time series features

Several properties of time series signals are exploited in this work for classification or regression tasks. They are defined in the following, and, if need arises, examples are given. Definitions are given for the univariate case for simplicity. For multivariate signals properties are extracted along the temporal axis for each sensor individually.

Minimum

The minimum x_{min} of a vector $\underline{x}$ is defined in Equation (3.3) [56, p. 60].

$$\forall y \in \underline{x} : y \geq x_{min} \tag{3.3}$$

Maximum

Vice versa to the minimum, the maximum x_{max} of $\underline{x}$ is defined in Equation (3.4) [56, p. 60].

$$\forall y \in \underline{x} : y \leq x_{max} \tag{3.4}$$

Mean

The sample mean $\overline{x}$ of a time series $\underline{x}$ is defined as the sum of all values x_i along the temporal axis divided by the number of measurements N (see Equation (3.5)[108, p. 36]).

$$\overline{x} = \frac{1}{N} \sum_{i=0}^{N-1} x_i \tag{3.5}$$

Mean Squared Deviation from the Mean Value

The Mean Squared Deviation from the Mean Value (MSDMV) for a time series represents the variation of measured values around their mean. We define the MSDMV as:

$$\mathrm{MSDMV}(\underline{x}) = \frac{1}{N} \sum_{i=0}^{N-1} (x_i - \overline{x})^2. \tag{3.6}$$

Root Mean Squared Deviation from the Mean Value

We define the Root Mean Squared Deviation from the Mean Value (RMSDMV) as the square root of the MSDMV. Combined with Equation (3.6), this results in Equation (3.7).

$$\mathrm{RMSDMV}(\underline{x}) = \sqrt{\mathrm{MSDMV}(\underline{x})} = \sqrt{\frac{1}{N} \sum_{i=0}^{N-1} (x_i - \overline{x})^2} \tag{3.7}$$

Skewness

Skewness describes the asymmetry of a distribution with respect to its mean. It is calculated via the Fisher-Pearson coefficient g_1 (see Equation (3.8) [162, p. 18]).

$$g_1 = \frac{\frac{1}{N}\sum_{i=0}^{N-1}(x_i - \overline{x})^3}{\sqrt{\frac{1}{N}\sum_{i=0}^{N-1}(x_i - \overline{x})^2}^{\,3}} \tag{3.8}$$

Kurtosis

The kurtosis describes the curvature of a distribution. When compared to the kurtosis of a normal distribution by subtracting 3, it is also refered to as excess. It can be calculated using Equation (3.9) [162, p. 19].

$$g_2 = \frac{1}{N} \sum_{i=0}^{N-1} \left(\frac{x_i - \overline{x}}{s} \right)^4 - 3 \tag{3.9}$$

Median

For a given dataset, its median is the value for which $50\,\%$ of the values are greater and $50\,\%$ are smaller. Therefore, it splits the dataset in two halfs. So, for a sorted vector $\underline{x}_{\text{sorted}}$ of length N, and $m = 1, 2, \ldots, N$, the median x_{median} is defined in Equation (3.10) [36, p. 27].

$$x_{\text{median}} = \begin{cases} \underline{x}_{\text{sorted}}\left[\frac{N-1}{2}\right], & \text{if N = 2m + 1} \\ \frac{1}{2} \cdot \left(\underline{x}_{\text{sorted}}\left[\frac{N}{2}\right] + \underline{x}_{\text{sorted}}\left[\frac{N}{2}+1\right] \right), & \text{if N = 2m.} \end{cases} \tag{3.10}$$

Quantile

A p-quantile x_p divides a distribution P according to Equation (3.11) [135, p. 524]. In this work, P is composed of all samples x_i of a input time series $\underline{x}$.

$$P((-\infty, x_p]) \geq p \quad \text{and} \quad P([x_p, +\infty)) \geq 1 - p \tag{3.11}$$

For $p = 0.5$ this matches the median. In the following features called “Quantile25” and “Quantile75” are used. The number 25 and 75 represent a p value of 0.25 and 0.75 respectively.

Energy

The energy of a digital signal is defined in Equation (3.12) [123, p. 144].

$$E(x) = \sum_{i=0}^{N-1} x_i^2 \tag{3.12}$$

Power

The power of a digital signal is defined in Equation (3.13) [98, p. 17].

$$P(x) = \frac{1}{N} \sum_{i=0}^{N-1} x_i^2 \tag{3.13}$$

It is the signal energy scaled by the signal length N.

Mean Absolute Deviation

The Mean Absolute Deviation (MAD) is similar to the standard deviation a representation of the variabilty of a dataset. However, it is more robust to outliers. Its definition is stated in Equation (3.14) [36, p. 33].

$$\text{MAD}(\underline{x}) = \frac{1}{N} \sum_{i=0}^{N-1} (x_i - \overline{x}) \tag{3.14}$$

Proportion of Positive Values

Dempster [34] introduces a new statistic called Proportion of Positive Values (PPV). For a given sequence $\underline{x}$, it describes the number of positive values divided by the sequence length N:

$$\text{PPV}(\underline{x}) = \frac{1}{N} \sum_{i=0}^{N-1} [x_i > 0]_I \,. \tag{3.15}$$

$[x_i > 0]_I$ is the Iverson bracket [50], where

$$[P]_I = \begin{cases} 1, & \text{for P is true} \\ 0, & \text{otherwise.} \end{cases} \tag{3.16}$$

As an example for PPV: In the case of an ideal sine wave the PPV would be close to 0.5 as half a period has positive values.

Number of Peaks

The number of peaks (NbrPeaks) is a feature devised for the time series classification in Chapter 5. The absence of peaks would often coincide with only one vehicle crossing the bridge, as shown in the leftmost plot of Figure 3.1. The center plot gives an example of at least two vehicles which cross the bridge with some distance, producing a distinct peak in the displacement signal. Finally, a high number of peaks can indicate the presence of bridge oscillation. Ultimately, the hyperparameters of this feature might need tuning for different structures.

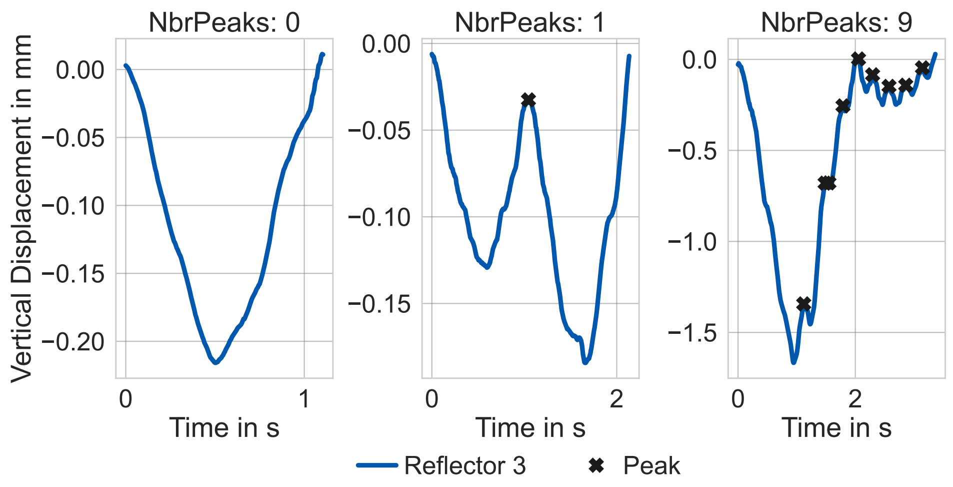


Fig. 3.1.: Three examples for nbrPeaks from Bridge A.

We determine it according to

$$\mathrm{NbrPeaks}(\underline{x}) = \mathrm{len}\left(\mathrm{find\_peaks}\left(\underline{x}, \mathrm{distance} = 4, \mathrm{width} = 5, \mathrm{rel\_height} = 0.5\right)\right), \tag{3.17}$$

where find_peaks is a peak finding method from the Python package `scpiy` [152] and len returns the number of found peaks. All hyperparameters in Equation (3.17) have been tuned manually via Bridge A and Bridge B data.

xMinPosRatio

"xMinPosRatio" is another self-developed feature which describes the position of the time series minimum scaled by the sequence length N:

$$\mathrm{xMinPosRatio}(\underline{x}) = \frac{1}{N} \cdot \mathrm{argmin}\left(\underline{x}\right). \tag{3.18}$$

$\mathrm{argmin}\left(\underline{\mathrm{x}}\right)$ returns the position of the minimum of the vector $\underline{x}$.
Minima for single events, e.g. events with only one vehicle on the bridge, tend to have their lowest value approximately in the center of their displacement time series. Conversely, an event involving several vehicles, like a truck followed by a car or vice versa, might have the minimum relatively early or late in the sequence. By scaling

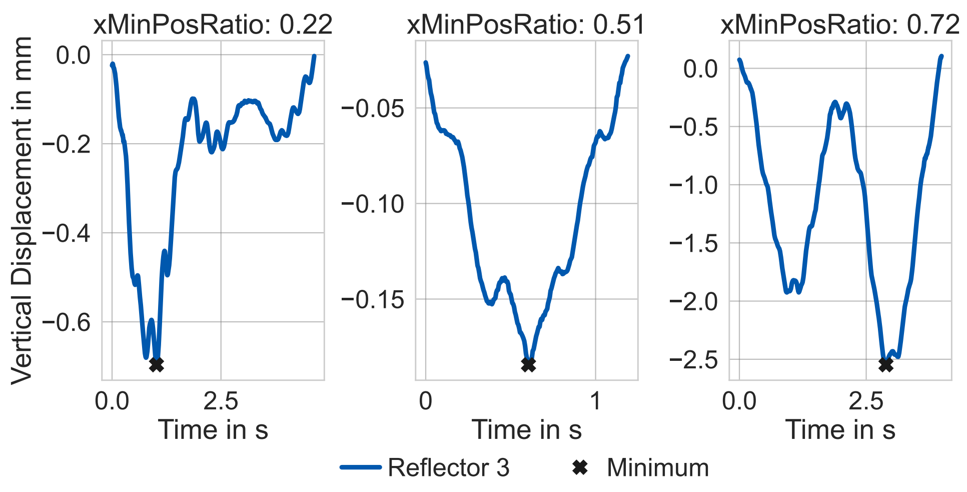


Fig. 3.2.: Three examples for xMinPosRatio from Bridge A.

with the sequence length, this parameter becomes independent of the same, making it more comparable over all instances. Thus, this feature could help to discern single and multi events. Figure 3.2 depicts three examples using reflector 3 from Bridge A. The first and last plot show events with several vehicles, the centered plot a single car crossing. On top of each plot, the calculated xMinPosRatio stated.

GradMax

GradMax stands for the maximum of the gradient of a time series $\underline{x}$. For more information on the gradient or derivative details, we refer the interested reader to e.g. [124]. Roughly speaking, the gradient of a signal descibes how fast it is changing. In this work, we rely on the gradient-function implementation of the Python-package `numpy`. Thus, GradMax is calculated as follows:

$$\mathrm{GradMax}(\underline{x}) = \max\left(\mathrm{gradient}\left(\underline{x}\right)\right) \tag{3.19}$$

Figuratively, a high GradMax value pinpoints towards a rapid displacement of the bridge which can correlate with a fast and/or heavy vehicle.

3.1.2 Infinite Impulse Response Filter

An Infinite Impulse Response (IIR) filter has coefficients a_k and b_k, so that

$$x_n^{out} = \sum_{k=0}^{q} b_k x_{n-k}^{in} - \sum_{k=1}^{p} a_k x_{n-k}^{out}, \tag{3.20}$$

where x^{in} is the input signal, x^{out} is the output signal and p and q depend on the filter order [123, p. 114]. Multiple techniques to calculate filter coefficients exits. One popular filter design is called Butterworth-Filter [19]. It is characterized by a plain passband. When combining an IIR filter in a forwards-backwards manner they have zero phase. In other words, they do not shift the output signal. As a trade-off, the backwards filtering requires an offline application of the filter.

3.1.3 Continuous Wavelet Transform

We will only superficially explain the concept of wavelets. For a more in-depth explanation, please see, e.g., [69]. One motivation behind wavelets is to acquire local frequency information while maintaining a high temporal resolution [...]. The method is analog, as the original signal is expressed as a family of functions. These functions are constructed from a so-called mother wavelet [123, p. 156]:

$$W(a,b) = \int_{-\infty}^{+\infty} f(t) \frac{1}{\sqrt{a}} \psi\left(\frac{t-b}{a}\right) dt \qquad a \in \mathbb{R}^+, b \in \mathbb{R} \tag{3.21}$$

Different coefficients are generated from the input signal by varying scaling a and time delay b.

One example of a mother-wavelet is the Gaussian wavelet [116]

$$\psi_{GW}(t) = C \cdot exp(-t^2), \tag{3.22}$$

where C is an order-dependent normalization factor. Another mother-wavelet is the Gaussian derivative wavelet, the m-th order derivative of Equation (3.22), where m lies between 1 and 8. It can be calculated according to Equation (3.23) [78, p. 131].

$$\psi_{GDW}(t) = (-1)^{m/2} \frac{d^m}{dt^m} \left(e^{-t^2/2}\right), \tag{3.23}$$

The Gaussian derivative wavelet will be used in this study, as implemented by Lee et al. [83].

3.2 Machine Learning

ML is a very popular and still growing field, which offers a wide range of methods for almost any task. A rapid increase in computing power over the last decades has reinforced the rise of ML contributing to more and more complex solutions. In general, (supervized) ML concerns itself with learning a mapping function between input data and a target value. But ML also includes other aspects such as preprocessing and feature engineering. In this section, we will explain the small selection of ML methods which are relevant in this work. For a more detailed introduction, we refer the reader to e.g. [107].

3.2.1 Preprocessing

Standard Scaling

An important question before training is how to present the data to a model. For instance, some methods perform better when input data is scaled to a standard range. Others even require different features to be on the same scale for meaningful results. One technique is standard scaling, which removes the mean for each feature seperately and then divides by the respective standard deviation. Mathematically speaking the transformed feature y can be calculated according to

$$y = \frac{x - \overline{x}}{s}, \tag{3.24}$$

where x represents the input feature and $\overline{x}$ and s its mean and standard deviation, respectively [140, p. 367]. The transformed output y, thus, has a mean of zero and a standard deviation of one.

Principal Component Analysis

Oftentimes, models receive input features between which a relationship exists, i.e. they are correlated. A higher input dimensionality by adding redundant features with no additional information can become a problem for models. The abundance of input features is oftentimes refered to as the “curse of dimensionality” or Hughes phenomenon [63]. Principal Component Analysis (PCA) is a transformation technique often used for linear dimensionality reduction, while preserving information [140].

It tries to find a new data representation, so-called main components, while maintaining the maximum of the dataset variance. This is achieved in an iterative procedure where every found component describes less variance. Ideally, the number of components greatly undercuts the input size without losing much information. The decision of how many components are outputted remains the responsibility of the user as a part of feature engineering.

3.2.2 Shallow Learning

One of the main objectives of ML is to find a mapping between input data and the target variable. Shallow learning is a part of ML which exploits models without hidden layers (see Section 3.2.3) to this end. There exists a multitude of shallow learning approaches, some of which are described in the following.

Linear Regression

A Linear Regression (LR) determines the function of the coefficient $\underline{w}$ and an error term ϵ, such that

$$\underline{y} = \underline{w} \cdot \underline{x} + \epsilon. \tag{3.25}$$

$\underline{y}$ is the target vector and $\underline{x}$ the input vector [22, p. 60]. To this end, the residual sum of squares between $\underline{y}$ and the predicted values is minimized with respect to the coefficients $\underline{w}$. $\underline{w}$ can be expressed using Equation (3.26) [80, p. 108].

$$\underline{w} = \left(\underline{x}^T \underline{x}\right)^{-1} \underline{x}^T y. \tag{3.26}$$

Ridge

Ridge [61] minimizes a least square function with the ℓ_2-norm $||\cdot||_2$ as regularization. The objective function is

$$\min_{\underline{w}} ||\underline{xw} - y||_2^2 + \alpha||\underline{w}||_2^2, \tag{3.27}$$

where $\underline{x}$ is the input vector, y the target variable and $\underline{w}$ the trainable weights [134, p. 5]. α is a hyperparameter that controls the amount of shrinkage.

Logistic Regression

Logistic Regression calculates the probability for each observation x_i to belong to a class $y_j \in 0, ..., K$. It estimates the probability $P(y_j = k|\underline{x}_i)$ according to Equation (3.28) [130, p. 252].

$$P(y_j = k|\underline{x}_i) = \frac{\exp(\underline{x}_i \underline{w}_k + \underline{w}_{0,k})}{\sum_{l=0}^{K} \exp(\underline{x}_i \underline{w}_l + \underline{w}_{0,l})} \tag{3.28}$$

Support Vector Machine

A Support Vector Machine (SVM) [122] searches a decision boundary between classes which maximizes the margin around this boundary. These boundaries can be linear or extracted from kernels for more complex shapes.

Decision Trees

A decision tree [55] is a non-parametric model, which makes predictions by learning decision rules from the training set. Those rules are simple thresholds by means of which new data is classified after checking it against one or more rules. A decision tree, which only consists of one rule e.g. has a depth of one, is called decision stump.

Random Forest

A Random Forest (RF) [18] is an ensemble method consisting of a collection of decision trees. Each decision tree is trained on a random, independent subset of the training set. The size of the subset is a hyperparameter. The results of all decision trees are then averaged.

Extremely Randomized Trees

Extremely Randomized Trees (ET) [52] are also an ensemble of decision trees. The difference to RF lies in the way the subsets are generated. The splitting rule herein has more randomness than in a RF. This is done to reduce the variance of the model.

AdaBoost

An alternative to ensemble desicion trees is to use boosting. Boosting describes an algorithm to ensemble several weak learners, such as desicion stumps, to a strong learner. AdaBoost (AB) [45] uses multiple stumps which are trained iteratively and after each step, the weighting of the training samples is adapted according on previous classification results. The final classifier is composed of all stumps, and each stump gets a weighted say in the final prediction.

Gradient Boosting

As the name suggests, Gradient Boosting (GB) [47] also uses the boosting technique. However, GB does not only use decision stumps but allows more complex trees. After using the inital probability for each class, following trees build upon the residual error of all previous trees. Adding more trees can thus reduce the residual slowly, depending on the set learning rate.

k-Nearest-Neighbours

k-Nearest-Neighbours (KNN) [28] are non-parametric models in the sense that the model contains no parameters that need to be trained. Instead, input samples are simply saved during training and then, during prediction, the k-nearest neighbours are inspected concerning their class affiliation or their target value for regression. The distance measure is a hyperparameter as well as the number of neighbours k. In general, input normalization is necessary especially for data with different scales.

Self-Organizing Map

The Self-Organizing Map (SOM) [79] has been developed to imitate the human brain in a sense that similar input activates similar regions in a net. To this end, the input vector is mapped to a low-dimensional space, usually 2D, which thus allows for dimensionality reduction and visualization. This map is generated from a shallow Artificial Neural Network (ANN) [128] (see Section 3.2.3) by minimizing the distance between the input and the weight vectors. The best matching unit and its neighbors are then moved towards the input vector. After many iterations a SOM learns to represent the input in a lower dimension.

Minimally Random Convolutional Kernel Transform

Feature-based approaches require meaningful input for succesful predictions. Preconceived representations come often into play during data-mining. However, manually engineered features might not be the most suitable ones. Therefore, approaches for automatic feature extraction exist. Minimally Random Convolutional Kernel Transform (MiniRocket) [35], for instance, is a state-of-the-art approach in the context of time series classification and regression. It applies 9996 convolutional kernels to the input time series to generate features. The kernels have a fixed length of 9, and their weights are restricted to two values: $\alpha = -1$ and $\beta = 2$. While the kernels of its predecessor Random Convolutional Kernel Transform (ROCKET) are random, MiniRocket relies on a set of 84 fixed kernels. These kernels have been selected by the authors as they achieve a high accuracy during testing, but they point out that other sets might be equally useful. Dilation, or the spread of a kernel over the time series, lies within the range of $\lfloor 2^0 \rfloor$ and $\lfloor 2^{max} \rfloor$, where $\max = log_2(l_{input} - 1)/8$ and with l_{input} are the input length. Padding is fixed and alternates between no applied padding and zero padding. Biases are calculated based on the result of the convolution of randomly selected training examples, therefore being the only non-deterministic aspect of the MiniRocket approach. For more details regarding convolutions in ML, we refer the reader to Section 3.2.3. As a last transformation step, the PPV (see Equation (3.15)) is drawn from each convolution result. For classification or regression, Dempster et al. [35] recommend Ridge if the training set size is smaller than $10\,000$. Otherwise they suggest the usage of Logistic Regression. Since PPV is scale-invariant, no normalization is carried out between transformation and prediction. Accordingly, the input data does not need to be normalized overall. MiniRocket can handle multivariate time series data and time series data of variable length. The authors provide an implementation via the Python-library scikit-time [90].

3.2.3 Deep Learning

DL is a part of ML [32] focusing on networks which have at least one so-called hidden layer between the input and the output layer. Most layers consist at least partially out of neurons [96]. Figure 3.3 illustrates the functionality of a single neuron. A given input $\underline{x} = (x_1, \ldots, x_N)$ is weighted with a vector $\underline{w} = (w_1, \ldots, w_N)$. The results are added together, including a bias value w_B. The sum is passed into a activation function Φ [141] yielding the final output y. Many activation functions exist, such as Rectified Linear Unit (ReLU) as it introduces non-linearites to DL

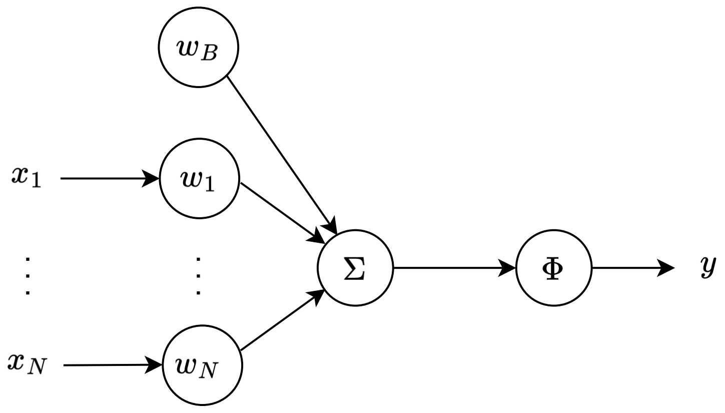


Fig. 3.3.: Functionality of a neuron: Input values are weighted with $w_1, \ldots, w_N$ and added together, including a bias value w_B. The sum is then passed into an activation function Φ.

models [53]. Another popular activation function is *tanh* which has an output range of $(-1, 1)$. In the output layer the "softmax" activation function is often used when facing classification tasks. It maps the input to a range of $(0, 1)$ and a sum of 1, simulating propabilities. The selection of the activation function falls to the developer as a part of the hyperparameter tuning. Weights, on the other hand, are learned by the model itself during training by backpropagation [84].

Artificial Neural Network

A layer that contains parallel neurons and in which each neuron is connected to all neurons of the previous layer, is called a Fully-Connected (FC) layer [2]. By stacking mutliple layers consecutively, complex functions can be modeled [132]. Models consisting entirely of FC layers are also refered to as deep feed-forward networks, or multi-layer perceptrons [96]. In the remainder of the work, we will refer to multi-layer perceptrons as ANNs. Figure 3.4 illustrates a simple ANN with one hidden layer. Each arrow symbolizes a weight learned during training. As all neurons of two consecutive layers are connected, the number of trainable weights increases rapidly with the size of the network. ANNs without a hidden layer can be refered to as shallow ANNs [128].
On a final note, the ouput $\underline{h}$ of a hidden layer is oftentimes refered to as a latent vector. It contains the features which are the representation of the input the model learned at this point of the network.

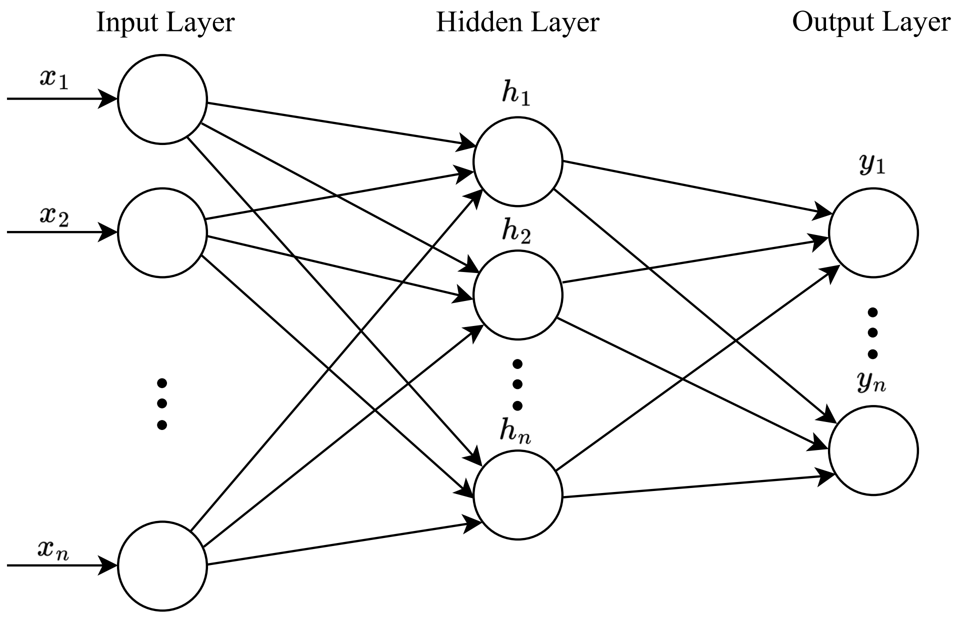


Fig. 3.4.: Example of an ANN with one hidden layer and multiple output values.

Convolutional Neural Network

Convolutional layers are an alternative to FC layers, exploiting convolutions as the name suggests. Practically, a convolutional layer convolves a group of kernels with the input data to generate feature maps [48]. During the training, the weights of a convolutional kernel and –depending on the settings– biases are learned. Several parameters need to be determined during tuning: *filter count, kernel length, dilation*, and *padding*. The filter count expresses the amount of convolutional filters that are applied in a layer. The kernel length is the length of the convolutional kernel, thus, it describes the number of trainable weights per filter. Dilation described the spread of a kernel. For a dilation of two, for instance, only every second input is used for convolution. Finally, padding determines if the input is padded (typically with zeros) to keep its initial length. Otherwise, no padding is applied, reducing the output size. One strength of Convolutional Neural Network (CNN) is that they have a comparably low number of trainable parameters as the kernels are slid over the input data [2]. This reduces training time as well as the required training set size. Furthermore, they do not require a specification of the input data shape. A CNN is usually composed of several consecutive convolutional layers as well as at least one FC layer. Many state-of-the-art DL approaches apply CNN for image related tasks, such as ResNet [57] and VGG [143]. Due to the similarities between images and time series data, several studies have investigated CNN for time series classification [42, 43].

3.2.4 Model Evaluation Metrics

Metrics are used in ML to evaluate the performance of a model but also to improve its performance by adapting weights and hyperparameters. A plethora of metrics exists, but we will focus on the ones used in this work. For classification tasks, we rely on Overall Accuracy (OA), Precision (P), Recall (RC) and Balanced Accuracy (BA). Regression tasks are evaluated using Mean Average Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and Coefficient of Determination (R^2).

Confusion Matrix

While it is strictly speaking no metric in itself, many classification metrics can be explained on basis of the confusion matrix. Furthermore, it grants fast insights into classification results. For a sample C_{ij} the confusion matrix states the ground-truth class i and the predicted class j. Its representation varies slightly depending on the conventions. This work adheres to scikit-learn [119]. The confusion matrix for a binary classification is displayed in Figure 3.5. In the case of the binary classification 4 cases can occur. True positive (TP) are correctly classified samples of the goal class. False positive (FP) on the other hand are misclassified samples of the same class. True negative (TN) and False negative (FN) are corresponding classification results of the second class.

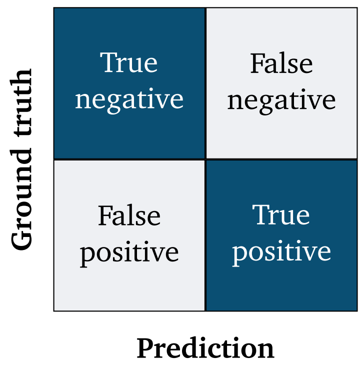


Fig. 3.5.: Confusion Matrix

Overall Accuracy

The OA is oftentimes used for performance analysis since it provides a clear understanding of the model performance as long as the dataset is balanced. OA describes how close predictions are to their actual value. For the binary case this yields

$$OA = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.29}$$

according to Rainio et al. [125]. For multi-class classifications the metric is calculated for each label and then the weighted mean is used [125].

Balanced Accuracy

For imbalanced datasets the BA gives a better understanding than OA, since takes all classes into account. For binary classification it can be calculated with Equation (3.30) [151].

$$BA = \frac{1}{2}\left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP}\right) \tag{3.30}$$

For multi-class classification the same rules apply as for OA.

Precision

The precision describes the capability of a model to not classify a negative sample as positive. For binary predictions, Rainio et al. [125] state it as:

$$P = \frac{TP}{TP + FP}. \tag{3.31}$$

For multi-class classification the same rules apply as for OA.

Recall

The recall score shows the models ability to classify positive samples as such. For binary predictions, it is defined as

$$RC = \frac{TP}{TP + FN}, \tag{3.32}$$

according to Rainio et al. [125]. For multi-class classification the same rules apply as for OA.

Mean Average Error

The MAE is defined as

$$MAE(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^{N-1} |y_i - \hat{y}_i| \tag{3.33}$$

according to Riese [126, p. 34]. It is the distance between the ground truth y and the prediction $\hat{y}$, which is also known as the ℓ_1-norm. A smaller value indicates a better model.

Mean Squared Error

As per Riese [126, p. 35], the MSE is defined as

$$MSE(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^{N-1} (y_i - \hat{y}_i)^2. \tag{3.34}$$

Compared to the MAE it amplifies the impact of outliers due to the squaring.

Root Mean Squared Error

The RMSE is defined as the square root of the MSE. Hence, it is calculated using Equation (3.35) [126, p. 35].

$$RMSE(y, \hat{y}) = \sqrt{MSE} = \sqrt{\frac{1}{N} \sum_{i=0}^{N-1} (y_i - \hat{y}_i)^2} \tag{3.35}$$

RMSE has the same unit as the target value, making it more intuitive than the MSE.

Coefficient of Determination

The R^2 represents the amount of the variance of the reference data y_i which is described by the prediciton $\hat{y}_i$. We use the scikit-learn [119] implementation, which is

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=0}^{N-1} (y_i - \hat{y}_i)^2}{\sum_{i=0}^{N-1} (y_i - \bar{y})^2}, \tag{3.36}$$

with $\bar{y} = \frac{1}{N}\sum_{i=0}^{N-1} y_i$ [154, p. 4]. A value close to 1 indicates good predictions. When having a value of 0.0 the model disregards the input features and just predicts the mean of y. With the scikit-learn implementation, negative values are also possible as $\sum_{i=0}^{N-1}(y_i - \hat{y}_i)^2$ can become arbitrarily worse. Although the R^2 is defined for linear models, it is a well established metric for nonlinear models in the conext of ML [3, 65, 112], as it can give a good intuition towards the models performance. As a stand-alone metric it can be misleading, which is why we also include other metrics such as the MSE in our evaluations.

3.2.5 Explainable AI

ML can solve and automate many difficult tasks. Yet, the underlying reasons for predictions are oftentimes unclear [131]. Especially complex ML models are a black box to both developers and users, raising legitimate doubts regarding their predictions. In the context of Structural Health Montoring (SHM), for instance, it needs to be explained why a model would make a certain prediction regarding a bridge condition. Thus, understanding the decision process is crucial to increase the trust towards ML applications. Explainable AI is the domain of AI which concerns itself with the aspect of explainability. On the on hand, we perform manual analysis of feature importances [87] and latent features in this thesis. Apart from that, more sophisticated methods such as Shapley Additive Explanations (SHAP) [88] are applied.

Feature importance

An advantage of tree-based classifiers such as RF is that they contain information about the importance of input features [87]. They are derived from the depth or rank of a feature in a decision tree. Averaging over all trees, can give a relative importance. However, the importance of features with a higher cardinality can be overestimated.

Shapley Additive Explanations

SHAP is an ad-hoc model-agnostic method of Explainable AI. It provides a unified approach to understand the model's decision process for specific input data [89]. SHAP is derived from Shapley values which stem from cooperative game theory and try to estimate the surplus of players generated in coalitions. A Shapley value is calculated by constructing all coalition variation and imputing the effect of each player. Transferred to ML, features represent the players and the surplus the amount a feature contributes towards the prediction. Due to coalition building, computation time grows exponentially with the number of features. This limits the original Shapley approach to simple models.

SHAP approximates Shapley values by sampling coalitions z', e.g. not using all possible coalitions, predicting on these coalitions and then fitting a weighted LR model on the samples [88]. The coefficients ϕ_i of the LR model represent the Shapley values for each feature i. Equation (3.37) [89, p. 6] expresses, how SHAP calculates Shapley values for a given model f and an input data point x. For tabular data, x represents one row. x' is the simplified data input. In computer vision, for instance, not every single pixel should be considered. Instead they are aggregated to superpixels x'. From x' a subset z' is drawn where features are either present or absent. In other words, features are turned on and off. One way to achieve this, is by replacing absent features by random values from the training dataset. Then, for z' the weighting parameter $\pi_x(z')$ is determined (see Equation (3.37)). M represents herein the maximum coalition size and $|z'|$ the current coalition size. $\pi_x(z')$ weighs both subsets with many present features, as well as subsets with many absent features highly. The idea behind this weighting is that with many features included, we can learn about feature interactions, whereas with few features, isolated effects are observed. z' is lastly passed to the model f once with and once without the feature i. The prediction outputs are subtracted returning the difference in prediction called marginal value. This is an advantage of SHAP as the output contribution is always in the same unit as the target variable. Totaling all marginal values returns the approximated Shapley value for a single input datapoint. The optimized Shapley values can be found by fitting a LR, minimizing the error of prediction between the target model f and the LR model. Another Shapley approximation approach is optimized towards tree-based models (TreeShap) [88].

$$\phi_i(f,x) = \sum_{z' \subseteq x'} \underbrace{\frac{M-1}{\binom{M}{|z'|}|z'|\,(M-|z'|)}}_{\pi_x(z')}[f_x(z') - f_x(z' \setminus i)] \tag{3.37}$$

One premise for SHAP to work accurately is that features have to be independent. Now in real-world data, this condition is rarely fullfilled. Feature correlation, for instance, can lead to attributing importance to irrelevant features. One solution is to model the coalition game using a different function, which can break dependencies [23]. But as a trade-off the model is evaluated on data points that lie outside the true data manifold. Consequently, there is no optimal solution. Chen et al. [23] frame this as a context dependent choice where one can either be true to the data in the sense that no out-of-distribution data is passed to the model, or be true to the model as all feature importances are stated correctly. With this trade-off in mind, SHAP can be a powerful tool to analyze ML models by investigating the impact of features on predictions. A Python implementation of SHAP is provided by Lundberg and Lee [89]. Unfortunately, at the time of writing their library cannot handle models which include preprocessing steps, or complex models such as MiniRocket.

Finally, to gain some intuition on how SHAP works, Figure 3.6 displays the waterfall plot of a toy example. In our example we are dealing with a binary classification task. A waterfall plot depicts the contribution of each feature towards the prediction on one single data point. Feature names are listed on the left and orderd according to their importance in this single case. For reasons of clarity, features with low importance are oftentimes combined at the bottom. As mentioned, an advantage of SHAP is that the individual contribution always has the same unit as the target value. So in our example, the output is the probability of a data point to belong to one of two classes. Accordingly, the x-axis range within $(0, 1)$. Starting point is the expected value, stated below the x-axis. Since the expected value is 0.85, we have an imbalanced dataset. The lowest features further contribute to a prediction of 1. However, “Feature 1” and “Feature 3” push the final probability to 0.31. The final prediction is stated above the end of the topmost contribution arrow.

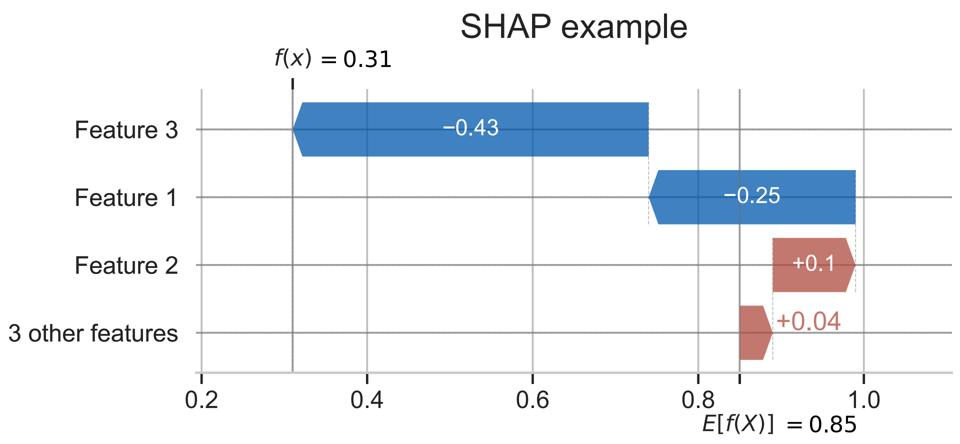


Fig. 3.6.: Waterfall plot of a toy example to showcase SHAP.

Part I

Vehicle Detection

Event Detection 4

Because it's only concrete and cars,
it's only sirens and missing stars,
it's only whiskey and disregard
in the smallest hours here, when I feel alone.

— **Ben Howard**
Singer/Songwriter

This chapter includes material from the following works:

M. Arnold and S. Keller. "Detection and Classification of Bridge Crossing Events With Gound-Based Interferometric Radar Data and Machine Learning Approaches". en. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* V-1-2020 (Aug. 2020), pp. 109–116

It is cited as [5] and marked with a orange line.

M. Arnold et al. "Convolutional Neural Networks for Detecting Bridge Crossing Events With Ground-Based Interferometric Radar Data". en. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* V-1-2021 (June 2021), pp. 31–38

It is cited as [4] and marked with a purple line.

4.1 Introduction

The overall objective of this thesis is to automatically identify vehicle configurations from GBR bridge displacement data using ML methods. For this purpose, it is necessary to detect vehicle crossings in the GBR data and extract the displacement time series of these crossings from start to end. In other words, the displacement time series needs to be segmented in sequences with and without vehicles on top of the bridge. Part I - Vehicle Detection deals with this research goal and describes the development and evaluation of such data-driven methods. Since labeled data is available, supervized ML methods are applied. Several sub-tasks have been defined with regard to the applicability of this event detection system. By applying as little preprocessing as possible to the measurement data, generalization over many infrastructures can be achieved. Developed algorithms should be independent of external influences such as environmental effects to further reduce preprocessing. Moreover, real-time capability has been attempted. In this context, real-time means detecting a vehicle while it crosses the bridge. Doing so can help validate predictions concerning traffic parameters as the prediction results are presented while vehicles are still in eyesight. Furthermore, when using GBR in other use cases, e.g. overweight vehicle enforcement or traffic control, fast evaluation is imperative. Finally, events extracted can be stored in a database more easily, as required in ZEBBRA, while discarding irrelevant data sections.

In order to achieve a segmentation fulfilling those requirements, we cannot process the entire time series in one batch. Therefore, we convert the time series segmentation task into a data-driven classification with a sliding window of $0.5\,\mathrm{s}$.

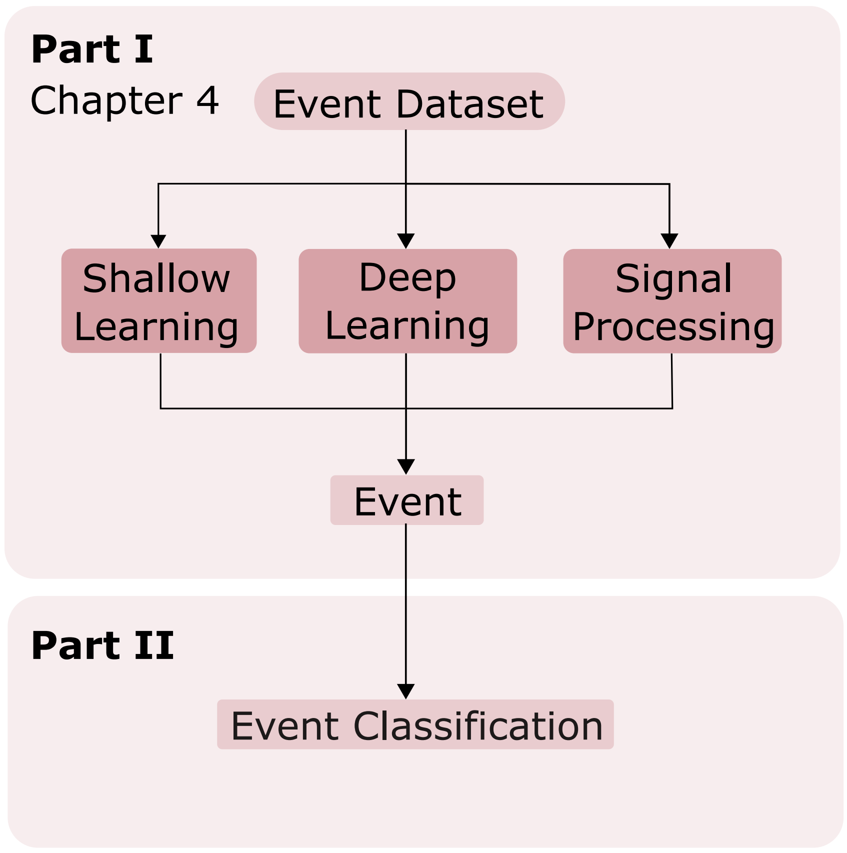


Fig. 4.1.: Overview of the event detection regarding databasis and methodology.

For each window we determine if a vehicle is currently on top of the bridge or not. Consequently, we lose context information as windows are treated individually. Three different classification approaches have been investigated: shallow learning, DL and signal processing. Figure 4.1 gives an overview of this chapter regarding data basis and methodology. The event dataset, which we use in this chapter, will be described in Section 4.3. Initially, we investigate shallow learners on a subset of the dataset for data-driven classification. Methodology and results are stated and discussed in Section 4.4. Afterwards, we evaluate a DL approach in Section 4.5. As mentioned, both sections focus on the classification task of individual sequences and neglect the segmentation of events as a whole. For this reason the best approach is evaluated on the basis of a full time series and compared to a simple thresholding baseline via classic signal processing in Section 4.6. The comparison is based on their capability to segment vehicle crossings from start to finish. In Section 4.7 an overall conclusion for the event detection is drawn. First of all, however, the following Section 4.2 gives an overview of relevant studies concerning time series classification and segmentation as well as event detection in different fields.

4.2 Related Work

In this section, we will first list state-of-the-art time series classification methods. Then, we will survey studies with similar classification or segmentation tasks from different fields. Finally, we describe the current state-of-the-art in vehicle detection in the context of bridge monitoring.

Classification of time series data plays a role in a variety of fields such as finance [139], economy [29], and biology [21, 25]. The UCR archive combines many different tasks to one collection, serving as a benchmark for new algorithms [30]. Yet, the datasets are small compared to common image databases. Accordingly, groundbreaking new approaches are rarely developed and new models are mostly recreations of famous image classification models.
In general, there are four types of time series classification methods: distance-based, feature-based, ensembles of the previous approaches, and DL-based. Dynamic Time Warping (DTW), for example, is a distance measure searching for the best path between two sequences. Combined with a KNN classifier, it oftentimes serves as a baseline [133]. While DTW can handle time series of variable length, it may fail for sequences with multiple peaks [138]. Another drawback of DTW is its high computation time, especially for long sequences. To an extend, this can be reduced by leveraging hyperparameters such as a window for path searching.

Another option for faster training is to reduce the data dimensionality, which is usually done by feature-based approaches. Certain characteristics, such as statistical quantities, are extracted from the time series data and then inputted in the model. These features represent global or local information. Their relevance can vary depending on the target value making manual data engineering a challenging task [49]. Several python packages exist which concentrate on time series feature extraction, offering a wide range of time series characteristics to choose from [27, 11]. Automatic data mining can reduce the human factor. Schäfer [138] introduces Bag-of-SFA-Symbols (BOSS), where SFA stands for Symbolic Fourier Approximation. They are quantized representations of subsequences of the input time series. SFA are counted and a newly proposed histogram-based distance is then used to compare sequences. Schäfer [137] extends BOSS to a more scalable version by compact representation of classes.
Another way to extract features involves random convolutional kernels for transformation. As these kernels do not have to be trained, the transformation process is very fast. Jarrett et al. [66] discover that random convolutional kernels can achieve good performances for image classifications tasks when the sample size is low. They state that random kernels can already represent oriented filters. However, they require rectification, pooling and normalization. As time series datasets are notoriously small, this approach has been investigated by several studies [44, 67, 41]. Farahmand et al. [41], for instance, propose random filter banks for time series transformation. In combination with linear models such as LR, they succesfully perform classifications on UCR datasets. ROCKET [34] and MiniRocket [35], the latter of which we detail in Section 3.2.2, are famous models exploiting this concept by applying almost 10 000 random kernels. ROCKET [34] is a famous model exploiting this concept by applying almost 10 000 random kernels. MiniRocket, which we detail in Section 3.2.2, replaces the random weights with a set of fixed values. Then, the PPV is drawn from the result of each convolution and inputted into a linear model for prediction. Thereby, it achieves state-of-the-art results on the UCR archive. Tan et al. [146] revise this concept to MultiRocket by reducing the number of kernels and adding three additional features per kernel.

A common problem of feature-based methods is their focus on a single property. Ensembles accomodate for this by fusing several heterogenous approaches into one. The Collective of Transformation-Based Ensembles (COTE) focuses on classifiers with different domain representations consisting of 35 classifiers overall [10]. Lines et al. [85] extend COTE to Hierarchical Vote Collective of Transformation-based Ensembles (HIVE-COTE), adding other models such as BOSS but also changing the voting to hierarchical voting. Middlehurst [103] further improves it to a new version Hierarchical Vote Collective of Transformation-based Ensembles 2.0 (HIVE-COTE 2.0) by, for example, adding ROCKET to the ensemble. While HIVE-COTE 2.0 performs well on the UCR archive, the abundance of contained methods slows down training tremendously and furthermore reduces understanding of its behaviour.

To avoid expensive manual data mining, DL has been investigated in time series classification recently [94, 160]. Wang et al. [153] create a DL baseline by testing three different models on the UCR archive. One is a simple Fully-Convolutional Network (FCN) and the other two mimic the famous Residual Network (ResNet). Fawaz et al. [42] compare nine different end-to-end DL approaches and show that the models of Wang et al. [153] outperform the others on most UCR datasets. They also highlight an advantage of certain FCN as to their predictive behaviour. It can be visualized using so-called Class Activation Map (CAM), making them more interpretable. Fawaz et al. [43] introduce InceptionTime. Its core, the Inception module, applies three convolutional filters with varying length as well as a bottleneck filter. InceptionTime performs comparable to HIVE-COTE 2.0 on the UCR archive with considerably lower training time. Usmankhujaev et al. [149] extend InceptionTime to InceptionFCN by introducing residual connections among other changes. This approach outperforms its predecessor on a majority of the UCR datasets and also reduces its network complexity. Other approaches try to leverage more on the sequential information of time series data. Recurrent Neural Network (RNN) utilize different implementations of a memory to preserve information while sliding over the input data. Initially designed for Natural Language Processing (NLP), they can also be used for time series tasks. Karim et al. [72] investigate a Long Short Term Memory (LSTM) RNN combined with a FCN. Tested on the UCR archive, their method outperforms the state-of-the-art approach on at least 43 datasets. However, LSTM introduce a big complexity overhead, which can lead to overfitting especially on small datasets. Finally, Transformers are an encoder-decoder framework also developed for NLP [150]. Zerveas et al. [159] transfer the concept to the time series domain by removing the decoder. In general, DL models are among the state-of-the-art competitors on the URC archive. Nevertheless as Fawaz et al. [43] state, there is a high risk of overfitting for small datasets. Additionally, many labeled data points are required for the training of DL methods.

Concerning event detection in time series data, an illustrative example is the detection of the heartbeat or the QRS complex in an Electrocardiogram (ECG) signal. Furthermore, the waveform has some resemblance to displacement measurements from bridge vehicle crossings. Unlike the latter, the duration of a heart beat is significantly shorter with up to $0.1\,\mathrm{s}$ and its duration is usually constant. DL, especially

CNN, are being investigated in this context [21, 13]. Silva et al. [142] use a CNN on 300 sample long windows to detect QRS complexes. They only label a window as a positive sample if a QRS complex is positioned in its middle. Real-time capability is also researched in some papers, e.g. by Chen et al. [24], using standard signal processing methods. He et al. [58] apply filtering and a LSTM for QRS complex detection. Again, the label is assigned according to the complex position. For a subsequent analysis simply detecting a complex is not enough. Instead, the aim is the extraction of the whole cardiac cycle via segmentation. Here, approaches based on filtering and thresholds are still state-of-the-art [155, 117]. Moskalenko et al. [110] train a CNN with U-net architecture to segment 10 s long ECG signals in four classes. With the same goal, Matias et al. [95] test a CNN on long ECG signals with as DTW as baseline. They show that the CNN performs better, however, they do not investigate real-time capability.
From a different perspective, events might also be considered anomalies in a sense that they are deviations from the default state. Anomaly detection in time series data is a current issue in many different fields [26]. The idea is to train models on the normal conditions by letting them predict future values or reconstruct the input. Then, after deployment the prediction error is used as a criterion to detect anomalies. Munir et al. [111], for instance, test a CNN to this end on several benchmark datasets. These anomalies are oftentimes single outliers in time series data and not so much events of longer duration.

Finally, in the context of bridge traffic monitoring, most studies focus on detecting axles individually as they are tantamount to vehicle crossings [156]. With an axle detection sensor on each side of the bridge a vehicle's entering and leaving can be registered. In this way, axle detection can take the role of a more sophisticated event segmentation. As we cover axle detection more detailed in Chapter 6, we will only give a broad overview of studies which highlight the aspect of event detection or segmentation. In classical BWIM, special axle sensors are deployed on top of the bridge. Different sensor types have been utilized [158, 156]. Vehicle crossings can be easily extracted using simple thresholds as axles are prominent peaks in the measured signal. However, these implementations are subject to heavy loads and thus suffer damage quickly. And maintenance work can only be done by interrupting the traffic flow. Consequently, less delicate sensor positions are investigated. One solution is to attach e.g. strain sensors to the lower bridge side leading to Nothing-on-Road (NOR) BWIM. Despite the distance between sensor and vehicle, the axle configuration can be extracted from the signal using simple thresholds [70]. Ma et al. [91] dive into the topic of event detection from strain sensors, but again, they only apply a more sophisticated threshold-based approach. DL is only used in the subsequent steps of vehicle classification. Regarding bridge displacement, Michel and Keller [101] mention using a threshold to detect events from GBR signals. As their main goal is to find bridge oscillations, they do not address the topic of segmenting an event from beginning to end. They apply several preprocessing steps before the detection to remove longterm drifts and other disturbances [100]. These extensive steps prevent the application of this method from real-time detection, as some steps require long windows. Moreover, they filter out the bridge oscillations in its eigenfrequency using a bandpass filter impeding generalization over many structures. Another study which investigates bridge displacement data uses a camera to extract information about the presence of vehicles on the bridge [115].

Research Gap Overall, a gap currently exists regarding the detection of vehicles on bridges using bridge displacement data only. This gap comes as no surprise as easy-to-perform bridge displacement measurement methods such as GBRs have only emerged recently. This chapter aims at filling this gap. Studies on detection and segmentation in the context of ECG signals, which have a similiar waveform, show that ML can achieve satisfying results. However, the usage in real-time is rarely investigated, especially for segmentation. This can be useful when using GBRs for overweight vehicle enforcement or traffic control. We convert the segmentation task into a real-time classification task by classifying small parts of the displacement signal using a sliding window. For the classification we evaluate shallow learners as well as DL CNN.

4.3 Event Dataset

We define an **event** as the time during which a vehicle crosses the monitored field. In the case of Bridge A, this means that only one field, not the whole bridge, is considered. For lighter vehicles, such as passenger cars, no displacement can be measured if the vehicle is on the other field. It would be impossible in those cases to distinguish noise from vehicles being on the other field exploiting data-driven approaches. Figure 4.2 presents the vertical displacement during three events for all five reflectors at Bridge A (cf. Figure 2.3). The depicted situation involves several cars queueing behind a slower truck, which is commonplace. The truck's heavy weight causes a strong oscillation of the bridge, which slowly decays over several seconds. Vehicles entering during this decay still produce a bend of the bridge. However, the deformation is superimposed by vibration. Nevertheless, we only want to detect the bending process as an event, regardless of the general bridge oscillation, which is classified as a non-event. Extracting the shaded areas is the **Research Goal I** of this work.

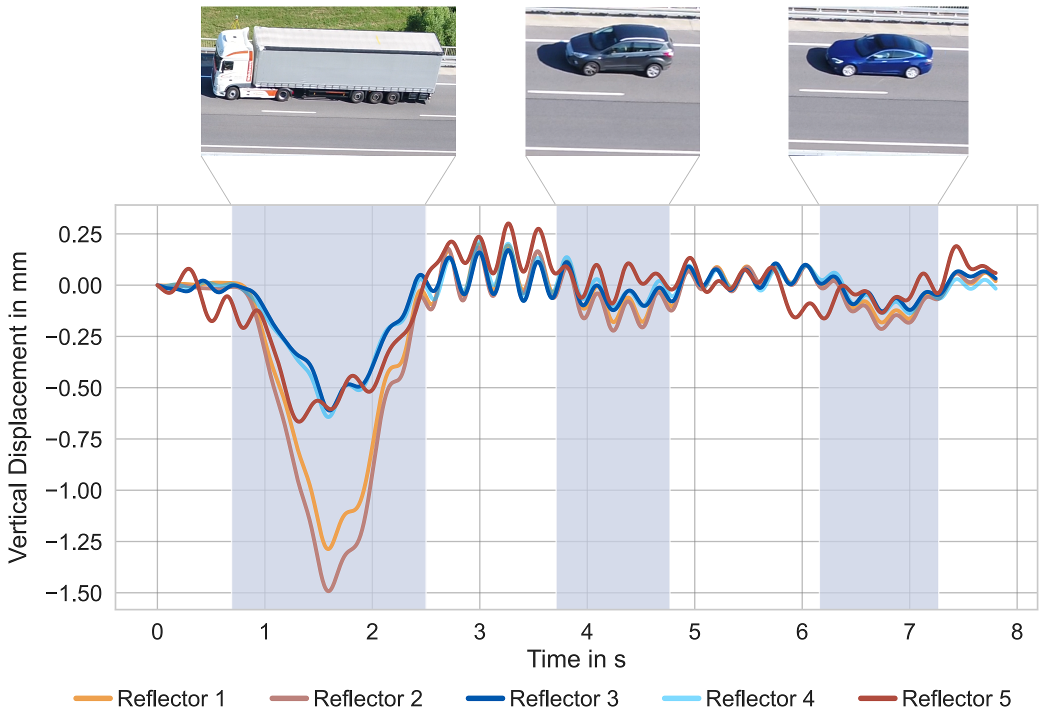


Fig. 4.2.: The displacement time series of three consecutive events for all reflectors on Bridge A. All events are highlighted and pictures of the corresponding vehicles, acquired with a UAV, are attached. The leading truck causes the bridge to vibrate even after leaving the bridge deck. Both following cars lead to a bending of the bridge which is superimposed with the oscillation. Adapted from [4] by changing the color palette.

To this end, the time series displacement of several measurement campaigns have been labelled by experts. We have implemented a Graphical User Interface (GUI) in Python to facilitate the labeling process. Table 4.1 states the figures of events for five measurement campaigns conducted during different days, including the event figures per day. As an event takes approximately $1.35\,\mathrm{s}$ on average, the number of non-event incidents is much larger considering each campaign's duration. The number of range bins refers to how many range bins we used for training. They have been selected following their respective SNR to ensure low noise. Since the bridge's eigenfrequency generally varies depending on the air temperature [93], we ensure a wide temperature range when selecting the study's dataset. The rows 1 to 4 in Table 4.1 are combined into a **dataset I**, which is partially used for the training of the models. Besides, row 5 serves as an entirely unknown dataset (**dataset II**) to specifically evaluate the performance in distinguishing decays and events.

Tab. 4.1.: Overview of the measurement campaigns at the bridges conducted for this study. The duration refers to the period of measurements at the respective bridge, number of events to the number of crossing vehicles, and number of range bins to the number of bins selected for analysis. The temperature range covers the air temperature during the measurements. Dataset I consists of the training, validation, and test subsets necessary for the ML approaches; while dataset II is an extra test subset, which is entirely unknown to the ML approaches. Adapted from [4]

Nbr.	Bridge	Duration	Number of events	Number of range bins	Temperature range	Dataset
1	A	4h30min	1 007	6	11.20 °C to 12.35 °C	I
2	A	1h45min	395	5	8.90 °C to 10.0 °C	I
3	A	30min	119	2	8.10 °C to 8.90 °C	I
4	B	3h30min	1 482	2	14.70 °C to 15.50 °C	I
5	A	1h20min	236	1	18.10 °C to 19.40 °C	II

4.4 Shallow Learning

For a preliminary test on the feasibility of a purely data-driven approach, a subset of the dataset was used. Several shallow learners as well as an ANN have been evaluated on this subset. To this end, seven features have been extracted from 0.5 s snippets of GBR time series signals using expert knowledge in the signal processing domain. The ANN is technically not a shallow learner. As it was trained on the same data and the same features, it is nevertheless included in this section.

4.4.1 Methodology

Our proposed methodological approach consists of three steps: the different preprocessing approaches, the feature extraction and the ML models to classify the events. Figure 4.3 provides the schema of the three applied steps.

Preprocessing

The GBR monitors a bridge dynamically with a sampling frequency of up to 200 Hz. However, most significant signal components of bridges are to be expected in a lower frequency range [97]. Therefore, low-pass (LP) filtering is a useful preprocesssing step to suppress high-frequency noise. We apply a Butterworth filter [16] with two different settings which we refer to as LP1 and LP2. The distinction between LP1

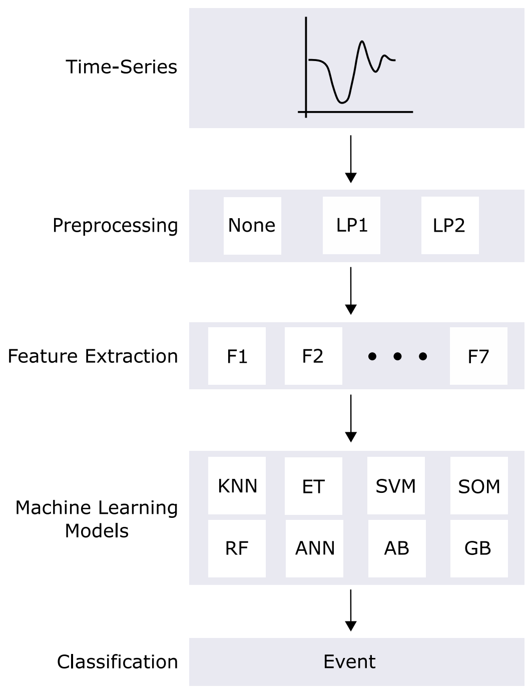


Fig. 4.3.: Methodology of the shallow learner approach. Adopted from [5].

and LP2 is that the latter has a higher degree of filtering than LP1. The raw GBR data without any filtering is defined as baseline.

We analyse different approaches for preprocessing (see Figure 4.3: None, LP1, LP2). In the test phase of the ML models we compare the results of the classification based on features extracted from the three preprocessing approaches. Note that no scaling has been applied on the time series and the features in order to avoid misclassification of events including heavy vehicles. [...]

Feature Extraction

The event duration is not constant. Since the further steps of our methodological approach require features extracted from time-series data of equal length, all events are split into multiple time series of a length of $0.5\,\mathrm{s}$. If at least $40\,\%$ of the $0.5\,\mathrm{s}$ window lie within the displacement caused by an event, the entire window is labeled as an event. The window size has been determined as part of the tuning process but outside of the grid-search. For each bridge, we consider only the range bin with the highest SNR. As a result, the complete dataset consists of 4490 time-series samples of $0.5\,\mathrm{s}$ for each preprocessing approach.

Tab. 4.2.: Seven features extracted from the GBR time-series data. The basis of calculation is given in Section 3.1.1. Adapted from [5].

Feature No.	Name of Feature
1	Maximum
2	Minimum
3	Energy
4	MSDMV
5	RMSDMV
6	Skewness
7	Kurtosis

Seven features are extracted from each time-series sample. We use common statistical and signal processing values such as [...] the signal energy. Since the displacement signal of the GBR has a baseline drift caused by environmental parameters (see Figure 2.2), we have excluded features such as the mean value which are sensitive to this drift. Where meaningful, e.g. *Maximum*, *Minimum* and *Energy*, the influence of the drift was eliminated [by subtracting the mean of the $0.5\,\mathrm{s}$ window prior to the calculation of the feature]. All the features are extracted directly from the time series (time domain) without performing a modal analysis. The extracted features are summarised in Table 4.2.

Machine Learning Models

In the last step (see Figure 4.3), we select eight ML models to distinguish between event and no event. We apply the following ML-Models for the classification task: KNN, ET, SVM, SOM[1], RF, ANN, AB and GB.

Before the training the complete dataset is randomly split into a training subset and test subset in the ratio $80{:}20$. All selected ML models are trained on the training subset using the extracted features as input and the class labels as target label. The hyperparameters for each model are found using grid search. Note that only the SOM performs the training phase unsupervised while all other models are supervised learners. For details on the SOM specifications see Riese [126].

During the subsequent test phase, the trained classification models classify the events meaning event vs. no event. The predicted classes are compared to the labelled values. The classification performance is expressed by the OA, P and RC. Ensemble

[1]Implementation from Riese [127]

models such as ET and AB provide the feature importance of the input features as further information of the classification task.

4.4.2 Results and Discussion

The main objective of this study is to evaluate the potential of ML models to detect and classify events based on GBR data from bridge monitoring. Such events are, for example, vehicle crossings on the bridge.

Table 4.3 shows the results for the event vs. no event classification task of the selected ML models and the three preprocessing approaches. When comparing the baseline, i.e. no applied preprocessing and hence more high frequency noise, to the LP1 and LP2 filtered preprocessing approach, we obtain a better classification performance in terms of OA. By applying the two LP filters as preprocessing, the eight ML models perform differently. Within the ensemble methods, ET, for example, achieves the highest OA for the unfiltered input data. The filtered input data leads to a lower OA. In contrast, RF performs the best with the most filtered input data (OA of $83.2\,\%$) as well as AB with an OA of $83.8\,\%$. With focus on P and RC, both measures are low despite OAs over $70\,\%$. The reasons for this is the unbalanced input data.

In sum, the classification performance of the ML models is satisfying. The best classification performance independent of the classification tasks is an OA of $83.8\,\%$. We cannot create further features out of the GBR data due to the included drift. Thus, we need to apply advanced preprocessing approaches to eliminate this drift and to extract more potential features. Also note that the impact of further features in the ML classification tasks needs to be evaluated with respect to each model. To allow for an unknown weight range, we have omitted data scaling which can be another factor influencing the prediction accuracy (see Section 4.4.1). Obviously, the classification performance highly depends on the selected time-series data. For example, we may miss the start of an event, if the impact of the increased bending of the bridge is too small to be detected in the respective window. To avoid these kind of misclassifications, we plan to use overlapping windows in further studies.

To summarise and visualise our results, we apply the two ML models performing the best with LP1 preprocessing on the event detection task (RF and AB) to classify a time series which is previously unknown by these two models. We then combine these two models in a voting classifier implementing soft voting to detect events (see Figure 4.4). By employing data of an unknown range bin with a different

Tab. 4.3.: Overview of the results of event vs. no event classification for the ML models under consideration. The classification performance is expressed by the OA, P and RC. The highlighted figures represent the best classification results on the test set for each preprocessing approach.

Model	Baseline			LP1			LP2		
	OA in %	P in %	RC in %	OA in %	P in %	RC in %	OA in %	P in %	RC in %
KNN	77.5	67.7	54.8	77.4	66.9	61.2	79.1	67.3	62.9
ET	**80.6**	67.6	66.0	70.0	64.7	18.7	76.9	72.5	37.8
SVM	72.4	67.0	24.4	59.2	36.5	33.0	76.6	75.4	72.7
SOM	56.9	35.9	47.0	55.6	35.5	43.9	57.8	37.8	61.5
RF	78.6	67.7	61.1	**82.2**	75.8	67.0	83.2	72.9	72.6
ANN	78.6	69.9	56.5	80.0	73.2	60.2	83.6	74.0	72.7
AB	78.2	67.7	59.4	81.6	74.9	66.0	**83.8**	74.0	71.6
GB	77.2	67.6	53.0	81.3	75.4	63.6	81.3	74.8	59.7

GBR position, we demonstrate the potential of the combination of ML approaches and GBR data to generalize in case of a event vs. no event classification. Bridges are dynamic infrastructure objects and vehicle crossings stimulate their oscillation. Hence, we have to deal with swing-out processes which are characterised partly by high amplitudes. As the classification of the events in general and the feature extraction is based on, for example, ratios between high amplitudes, the swing-out processes are challenging. At several occasions noise is falsely detected as an event, mirroring the precision of around $75.8\,\%$ and $74.9\,\%$ for RF and AB.

4.4.3 Conclusion and Outlook

In this section, we address the challenge of detecting vehicle crossings, which we refer to as events, and their classification based on GBR data using ML. In contrast to widely used event detection approaches, which exploit acceleration sensors or strain gauge data, we extract the bridge displacement directly from GBR time series data avoiding any threshold-based approaches. The GBR time series data has been recorded during real-world measurements at two bridges in Germany with random daily traffic. Our main objective is to investigate the potential of GBR data for detecting events.

To solve this classification task, we introduce a methodological approach involving an optional preprocessing step with LP filtering and eight ML models. The classification results reveal the high potential of data-driven models and the exploited GBR input

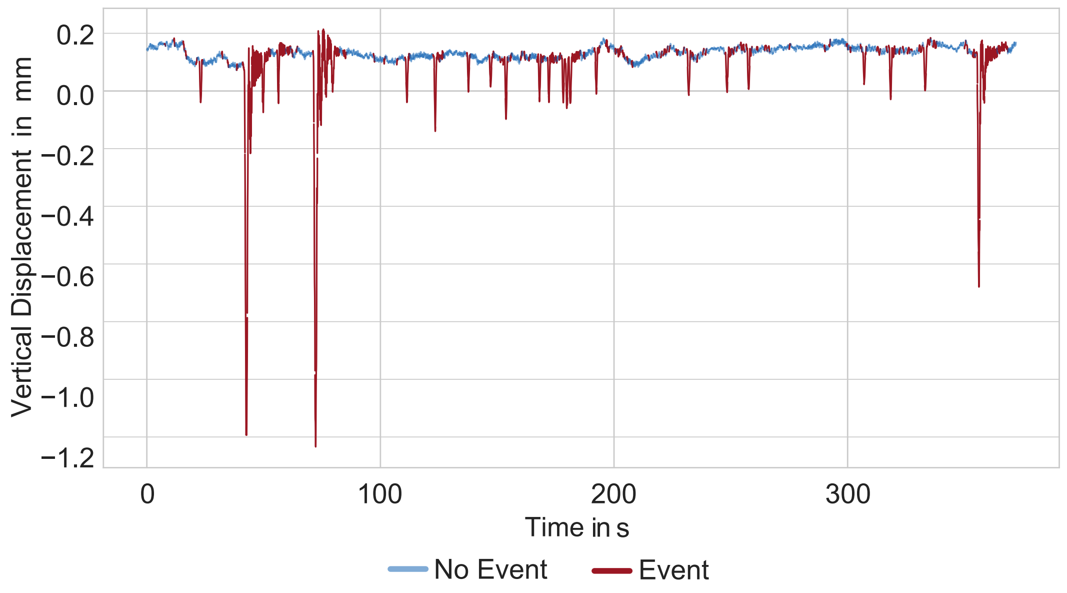


Fig. 4.4.: The vertical displacement of an unknown range bin at Bridge A and a different GBR position at this bridge as a time series over about 350 s. The event classification is shown as result of the voting classifier (RF and AB). Adapted from [5] by changing the color palette.

data at varying measurement conditions at the two bridges. Furthermore, the performance of the methodological approach is satisfying with respect to the limited size of (reference) data and without providing any prior domain-knowledge. Except for the LP filtering, no further preprocessing of the GBR data has been applied. Thus, the mentioned drift in the GBR time series data, caused, among others, by environmental influences during the measurements, has not been excluded in this preliminary study.

To evaluate the generalization abilities of the proposed approach, we apply a voting classifier (RF and AB) on time series data of a so far unknown range bin from data of another GBR sensor at Bridge A. Aiming at distinguishing between event and no event, the classification results are satisfying in general. Although, in some cases, noise is misclassified as an event, the events are all detected properly.

The results of this first study are very promising. As a direct consequence, we will further improve the ML-based approaches in order to remove the baseline drift. With respect to the class type classification of different vehicles, we will then be able to extract additional features from the time series GBR data. Then, we can evaluate the impact of these features on the classification performance due to avoiding misclassifications caused by long-term drifts. Furthermore, we consider to include a new class which covers the swing-out processes. To evaluate the generalization abilities in detail, we will expand the dataset in upcoming measurement campaigns. Additionally, we will exploit further range bins of each bridge. As the dataset size is increased, advanced ML approaches such as CNN can be applied and evaluated.

4.5 Deep Learning

Based on the results of Arnold and Keller [5], a follow up study was conducted to evaluate the potential of DL exploiting CNN for data-driven event detection [4]. A bigger dataset was used, since the labelling process could be streamlined using the shallow learners. More than one range bin per labeled time series was used as well. As a baseline a RF was trained as it was the best classifier for the intermediate filtering level.

4.5.1 Methodology

GBR Time Series Preprocessing

As an objective of this study, we apply as little data preprocessing as possible on the GBR-measured time series data. Based on this precaution, we will be able to deploy the proposed DL approach as an online tool to previously unmeasured bridges someday. Therefore, no normalization is applied before the training process of the ML models. Since bridges usually vibrate in a lower frequency range [97] than the GBR's Nyquist frequency of $100\,\mathrm{Hz}$, we use a LP Butterworth filter to remove high-frequency noise [16] as the only signal preprocessing step. The filter corresponds to LP1 in Section 4.4.1 and it is chosen as middle ground betweeen heavy filtering and no filtering at all. Additionally, we avoid removing the long-term drift to ensure the models' robustness against environmental impacts. The filtered time series data are automatically split based on non-overlapping windows of 100 samples, which means $0.5\,\mathrm{s}$, automatically. Subsequently, the data of each window is compared to a labeled event. If at least $40\,\%$ of the 100 samples window lie within the displacement caused by an event, the entire window is labeled as an event. This step is necessary for the preparation of the supervised classification task. Based on the defined window size of 100 samples and the ratio of $40\,\%$, we obtain a satisfying detection of the event's start- and endpoint. Besides, this window size ensures that we capture more than one period of the bridge vibration. The amount of events is significantly lower than the number of non-event incidences. This imbalance can influence the classification performance of the applied ML models since the entire dataset is biased towards non-events. Therefore, we apply a random undersampling by deleting samples of the majority class, which, in our case, is called a `no event`. As a last preprocessing step, we randomly split the dataset I with the ratio $70 : 15 : 15$ for the training, validation, and test subset. The training subset contains 47710 incidences overall with 23719 `event` and 23991 `no event`. Dataset II (see Section 4.3) is used as an entirely unknown test subset for the ML models. The `no event` class of dataset II solely consists of decays, which allows us a more thorough analysis concerning events and decays. It is composed of 236 events and 24 undisturbed decays, respectively 516 and 249 incidents, after splitting them into windows of 100 samples.

Machine Learning Models

We evaluate a solely data-driven DL approach for detecting events in GBR displacement data. As a baseline for the CNN classification performance, we rely on a feature-based ML approach, a RF model (see [5]). The RF is applied on features extracted from the LP filtered data such as the [... MSDMV] and implemented using the widely-used scikit-learn package [119]. The implementation of the introduced CNN is based on Tensorflow [1]. In the following, we describe the architecture of the (1D) CNN. As described in Section 3.2.3, CNN are mainly popular in image classification but are not often used in the context of 1D GBR time series data. The displacement time series of an event has similarities to the 1D grayscale gradient of an edge. Since edge detection is commonly performed by convoluting a filter, we exploit convolutional layers in our model. The CNN architecture is visualized in Figure 4.5. 100 bridge displacement samples of one single range bin represent the input. The next three parts consist of three 1D convolutional layers (CONV) with different filters and filter sizes, as summarized in Table 4.4. Each CONV layer is followed by a max-pooling layer (MaxPooling). The CONV1 layer has `tanh` as an activation function, in contrast to the other CONV layers. The main reason for this activation function is that the input data is not normalized to ensure the transferability to other, unseen bridge data. Based on the `tanh` activation, the first input features are mapped to the range -1 to 1. The additional CONV layers, therefore, have scaled input features. Relatively large `kernel_sizes` characterize CONV1 and CONV2, aiming at capturing gradients over several samples. As the GBR measures with a frequency of $200\,\text{Hz}$, the displacement change between consecutive samples is small. A large `kernel_size` allows the filter to register more significant changes within the signal. Additionally, the vibration of the signal is easier to detect, which ensures a more profound classification of pure decays and events superimposed with such vibrations. A flatten layer reshapes the CONV output to be used as an input for

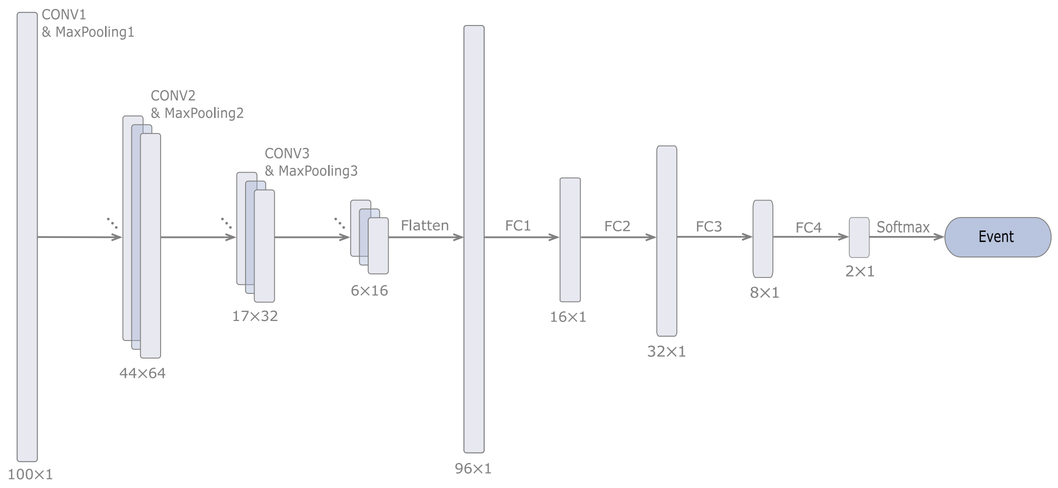


Fig. 4.5.: Flowchart of our developed CNN. The DL network combines convolutional layers (CONV), FC, and max-pooling (MaxPooling) layers. The output of each layer is presented as a vertical bar annotated with its corresponding shape. Details on the parameters of individual layers are given in Table 4.4. Adopted from [4].

FC layers. FC1 reduces the vector dimension, thus condensing on important features. Besides, FC2 expands, allowing for more complex evaluations based on the results of FC1. FC3 and FC4 then compress the features vector for a SoftMax, which returns the final classification results. The RF is implemented with the default hyperparameters except for the following parameters: `max_depth` $= 2$, `max_leaf_nodes` $= 6$, `min_samples_split` $= 0.6$ and `n_estimators` $= 200$. They are fine-tuned using GridSearch.

4.5.2 Results and Discussion

The study's objective is to investigate the potential of a DL approach in the context of detecting events in GBR time series data without any feature extraction. The classification performances of the RF with selected input features and of the CNN are contrasted in Figure 4.6 and Table 4.5. Figure 4.6 shows the normalized confusion matrix for the RF and the CNN model, while Table 4.5 states the OA, P, and RC. Both models are evaluated on the test subset of dataset I as well as on the entirely unknown dataset II (see Section 4.3).

In the case of **dataset I**, the RF and the CNN achieve high scores with values above $90\,\%$ (see Table 4.5 and Figure 4.6a). Besides, the CNN achieves a significantly better classification with an OA of $94.7\,\%$, which outperforms the RFs OA by 3.2 percentage points (p.p.). As dataset I contains events measured at two distinct bridges, this overall good performance reveals that the ML models can cope with heterogeneous

Tab. 4.4.: Hyperparameters of the applied CNN visualized in Figure 4.5. Adopted from [4].

Layer	Hyperparameters

CONV1	filters = 64
	kernel_size = 13
	activation = tanh

MaxPooling1	pool_size = 2

CONV2	filters = 32
	kernel_size = 11
	activation = ReLU

MaxPooling2	pool_size = 2

CONV3	filters = 16
	kernel_size = 5
	activation = ReLU

MaxPooling3	pool_size = 2
FC1	units = 16
FC2	units = 32
FC3	units = 8

GBR time series data. Therefore, the results indicate an appropriate transferability between data measured at different bridges (under specific prerequisites). When focusing on the correctly classified events concerning all events, the RC-score, the CNN, and the RF achieve almost equal scores. The CNN performance is only 0.8 p.p. better than the RF. However, the P-score differs considerably. The P-score indicates how many of the incidences classified as events are actual events. According to this score, the CNN ($95.3\,\%$) performs by 5 p.p. better. The main reason is that the RF experiences difficulties in distinguishing between decays and events, as shown in [5]. We have to consider that no information about the number of decay incidence of `no events` in the dataset I is given, which impedes a further and more detailed analysis.

Therefore, we focus mainly on the distinction between decays as `no events` and `events` in **dataset II**. The classification results based on this dataset reveal that the CNN outperforms the feature-based RF (see Figure 4.6b and Table 4.5). Although the OA decreases compared to the results achieved with dataset I, all metrics scores are still above $90\,\%$, with the RC for the CNN even at $99.4\,\%$. When focusing on the RF's performance, it strikes that the RF model achieves an almost equally high RC-score, but the P-score and especially the OA are relatively low. According to the confusion matrix in Figure 4.6b, the RF classifies the majority of `no event` decay

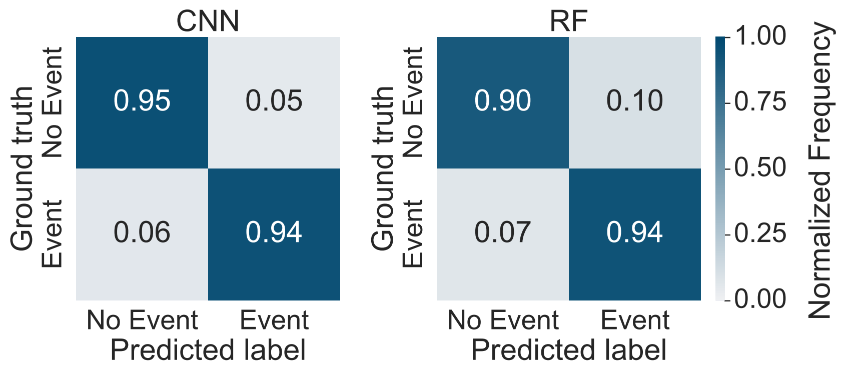
Tab. 4.5.: Results of the applied ML models, RF and CNN, for the dataset I and II on the classification task `event` vs. `no event`. Adopted from [4].

Model	Dataset I			Dataset II		
	OA in %	P in %	RC in %	OA in %	P in %	RC in %
RF	91.5	90.3	93.5	64.5	74.6	98.1
CNN	**94.7**	**95.3**	**94.3**	**92.7**	**93.6**	**99.4**

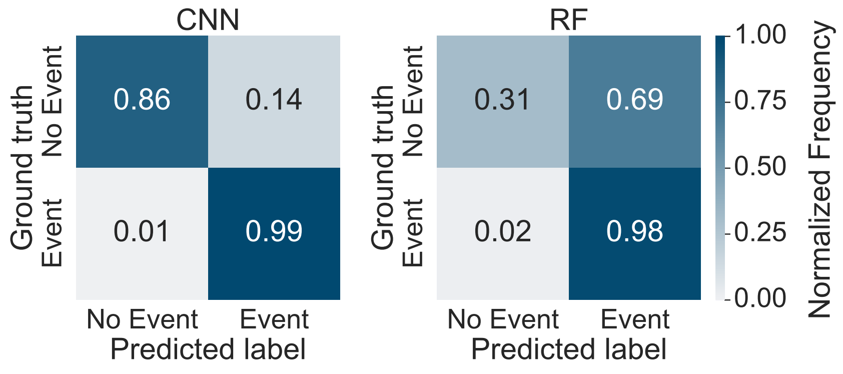
(69 %) as an `event`. This is again due to the challenges of distinguishing decays and events. Since in dataset II all `no events` consist of decays, the RF misclassifies many incidents. Taking the deviating atmospheric conditions included in dataset II from dataset I, we recognize that the CNN can handle the deviation in the air temperature range and the resulting drift in the GBR-based time series. Thus, our approach with a minimized preprocessing of the GBR time series data seems appropriate in applying a CNN. This allows us to exploit our model during measurement campaigns during different environmental conditions.

Figure 4.7 shows a classification example of a 60 s period of both the CNN and the RF model. Note that the shown period originates from the dataset II, which is entirely unknown to the ML models. The CNN can detect and distinguish events and decays. It only classifies a small proportion of the given excerpt falsely. One reason for this finding could be that the example excerpt, including an event's end, extends into the decay. As a solution, we can, for example, reduce the step size for the prediction window. The RF, on the other hand, fails in this case since it classifies the complete decay as an `event` after the first vehicle's crossing at around 3 s.

The window highlighted in pale blue corresponds to the vertical displacement situation visualized in Figure 4.2. In this situation, two cars cross the bridge shortly after a truck caused the bridge to vibrate. Thus, the bending caused by the two cars is superimposed by the bridge oscillation. It sticks out that the CNN performs better in extracting only the events regardless of the decay. This finding implies that the CNN learns to filter out the oscillation and considers the actual bending of the bridge. Based solely on the selected input features, the RF is incapable of classifying the entire excerpt correctly and considers the decay as an event. Both ML models have in common that they detect the single, undisturbed events at approximately 36 s and 52 s successfully. Although a small offset of around 0.25 mm caused, for example, by temperature changes is present, the ML models' performances are not affected.



(a)



(b)

Fig. 4.6.: Confusion matrices of the CNN model (left) and the RF model (right) for (a) dataset I and (b) dataset II. Adopted from [4].

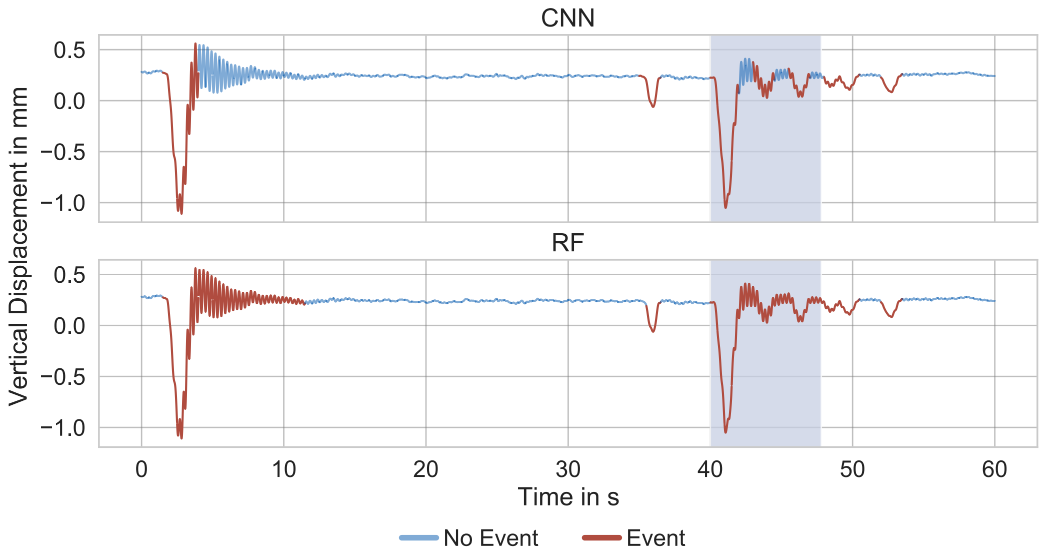


Fig. 4.7.: Classification results of an exemplary, 60 s long section of Bridge A for CNN (upper) and RF (lower). The measurement campaign from which this section originates is unknown to the model. The highlighted area corresponds to the excerpt depicted in Figure 4.2. Adapted from [4] by changing the color palette.

In [7], we indicate that the decay misclassifications of the RF can be leveraged. By combining both approaches, sequences classified as `no event` by the CNN and `event` by the RF are bridge oscillations. Determining their eigenfrequency can be used for automatic SHM [92]. Furthermore, this combination informs about events which are superimposed by bridge oscillations such as the car crossings at approximately 45 s in Figure 4.7.

In summary, our results show that a data-driven CNN is superior in event detection based on displacement time series as compared to feature-based models. While both approaches achieve appropriate results in general, the main difference is the capability of distinguishing events and decays. In this case, the CNN performs well, while the RF classifies a majority of the incidences incorrectly.

Feature Analysis

With the RFs of Section 4.4 and Section 4.5 we have two similar models trained on different but overlapping datasets. Hereon, a comparison of their feature importances can be conducted. We will refer to them as RF1 (Section 4.4) and RF2 (Section 4.5) from now on. RF1 witnessed a smaller dataset, which was also imbalanced. Figure 4.8 shows the feature importances for both RF. It shows that the importance of `Minimum` and `Maximum` only change slightly. While the `Kurtosis` has 0.0 in value, the `Skewness` is still very low. `Energy` and `MSDMV` are of major importance for RF1. Pressumably due to a bigger and balanced dataset, `Energy` equals `RMSDMV` in importance. The `MSDMV` has a smaller importance, too. However, since `MSDMV` and `RMSDMV` are completely correlated, this affects the calculation. Overall, the feature importances are similar showing that the features are robust over varying datasets.

Regarding explainable AI, it would be interesting to broadly understand the CNNs behaviour ad-hoc. One option herein is to see if there are correlations between its latent features and the features we extracted for our shallow learners (see Table 4.2). Figure 4.9 shows the scatter plot between two latent features and the `RMSDMV`. We will focus on these two as they have a very distinct behaviour. To extract them, we cut the CNN open after the first FC layer (FC1), leaving us with 16 latent features. Then, we input a time series unknown to the CNN and for each window we plot the latent features over the manual features calculated from the same windows. All datapoints are colored according to their predictions. Decays are identified by incorporating the RF as mentioned earlier: Cases in which the CNN does not detect an event but the RF does are regarded as bridge oscillations.

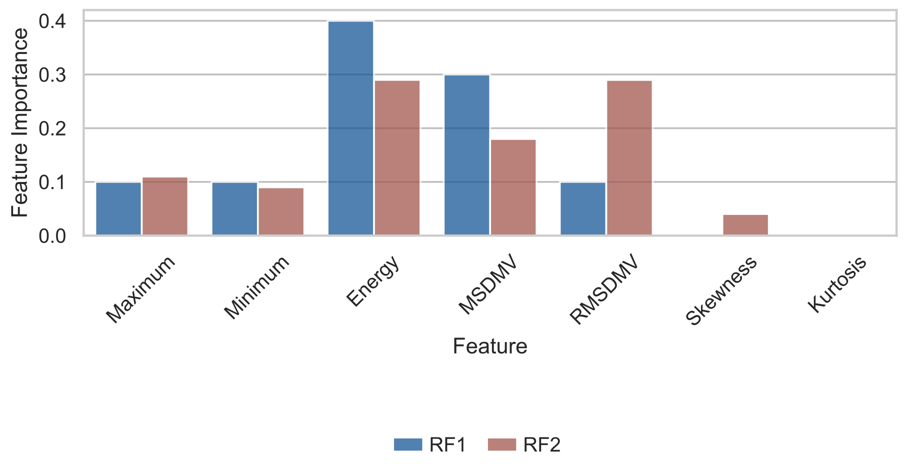


Fig. 4.8.: Features importances of RF1 and RF2.

Regarding this distinction, Figure 4.9 shows that those "Feature 1" and "Feature 5" of the CNN are close to zero for almost all "Decay" and "Non-Event" data points. In general, it became apparent that there was no clear boundary between their distributions for all CNN features. One reason might be that the convolutional layers already filter out the bridge eigenfrequency relaying no information of it to the subsequent FC layers. More importantly, one can easily spot a linear correlation between the RMSDMV and both CNN features, especially "Feature 5" for many data points. This is contrasted by many data points for which the feature value is zero. Speaking in ML terms, the corresponding neurons do not fire in some cases.

For further analysis, we plotted a part of the time series and highlighted windows for which the corresponding CNN feature returns values equal and unequal to zero (see Figure 4.10). Windows which are classified as an event are colored. The color depends on the feature value for this specific window. "Feature 1" is generally equal to zero if the window lies within the first half of an event. In contrast, it is unequal to zero in the second event half. Thus, "Feature 1" values unequal to zero coincide with upward slopes. The behaviour of "Feature 5" is inverted. Its neurons fire during a downward slope. Transferred to image processing, these two features represent filter results of edge detections once from bright to dark and vice versa. Interestingly, those CNN features also correlate with the RMSDMV if fired. It is to some extent intuitive as the RMSDMV is high in such cases. Ultimately, these results indicate that at least some parts of our CNN work in an explainable manner by learning one-dimensional edge detection filters and using latent features which correlate with common signal processing measures.

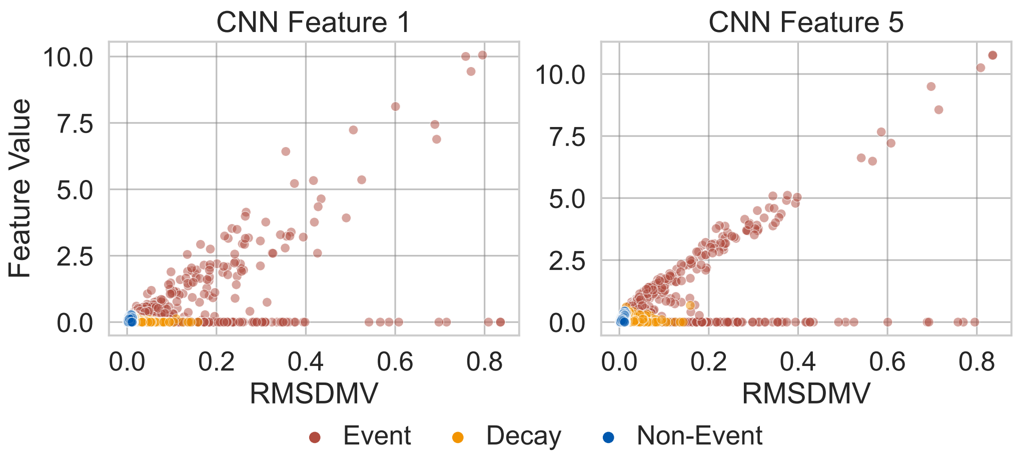


Fig. 4.9.: Scatter plot for two latent features of the first FC of our CNN over the `RMSDMV` of the same windows. The points are colored according to the prediction results.

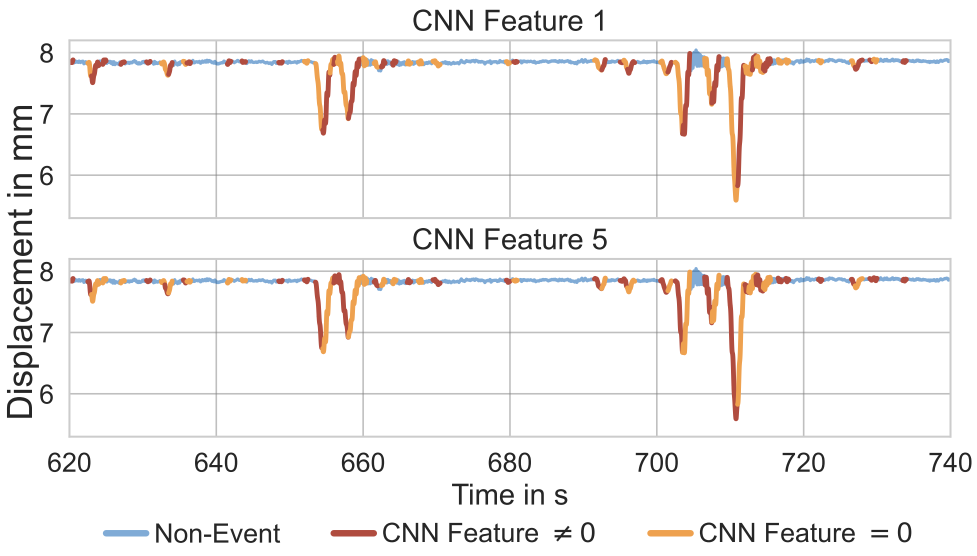


Fig. 4.10.: Example time series used for CNN feature analysis regarding both features investigated in Figure 4.9.

4.5.3 Conclusion and Outlook

In this section, we introduce and investigate a data-driven DL approach exploiting convolutional layers for event detection in 1D GBR-displacement time series data. Events refer to vehicles crossing the bridge during the GBR measurements. The data originates from real-world measurement campaigns at two German bridges. Our proposed DL approach addresses the current challenges, especially regarding the differentiation of crossing events and subsequent decays, present in state-of-the-art feature-based ML approaches based on shallow learners such as an RF model. The CNN detects events successfully and reliably; even in distinguishing decays and events, it performs satisfyingly. As a shallow baseline learner, we rely on a RF model. Overall, the CNN outperforms the RF in detecting only the bridge crossing while omitting the decay. In two distinct datasets, the ML models' performances are evaluated concerning, among others, the decays. The first dataset is used for training, validation, and test of the ML models. In contrast, the second dataset remains unknown in order to investigate the applied models' transferability and reproducibility.

Our CNN achieves higher overall OA, P, and RC on both datasets compared to the baseline model. The shortcomings of a feature-based, commonly applied ML model are revealed, especially in the second test on an unknown dataset. As presented, the CNN achieves an OA of $92.7\,\%$ and a P of $93.6\,\%$ in classifying events and decays (non-events). The RF falls short with an OA of $64.5\,\%$ and a P of $74.5\,\%$. Both models' RC-score is similar, implying that they rarely classify events as non-events. Besides, we compare the classification results exploiting a time series excerpt from a measurement campaign unexploited in the ML models' training. Both the CNN and the RF reveal robustness against environmental impacts since a long-term time series drift does not affect the classification performances. In order to improve the performance of the RF, gradient-based features could also be included in the future.

In sum, our study shows promising results for relying on a solely data-driven CNN as DL approach in the context of event detection with GBR-based time series data without any extensive preprocessing. Focusing on the further improvement of the CNN performance and a detailed evaluation of its limits and opportunities, we need to increase the number of decays in the training subset. Therefore, we will conduct further measurement campaigns at the presented bridges as well as add new bridges to the portfolio. Based on this new data, we can enhance the generalization abilities of our proposed DL approach. Besides, we need to consider including bridges with strongly different eigenfrequencies in our measurement campaigns to investigate the CNN performance concerning varying decay oscillations. Based on the presented DL classifiers, a bridge damage assessment could be established which functions under real-world conditions with non-invasive GBR time series data and stimulations caused by vehicle crossings. Such a prospective assessment could combine a data-driven event detection and subsequently an event classification to extract the input causing the measured bridge displacement. This combination might be a first step towards determining changes in the dynamics of bridges concerning a more profound structural health monitoring.

4.6 Event Segmentation

So far, all approaches have been evaluated using random excerpts from our dataset independent of their order. Concerning an end-to-end bridge event classification one important aspect of the event detection is to extract events from start to end such that it is sufficient for a follow-up analysis. Therefore, we evaluate our approach with respect to this capability in this section. As a baseline, we will use an adaptive threshold.

4.6.1 Data basis

We conduct this analysis on measurements from Bridge A and Bridge B, which are not part of the event dataset. The Bridge A sample is $1800\,\mathrm{s}$ long and is depicted in Figure 4.11. 101 events occur during the time span with an average duration of $1.54\,\mathrm{s}$. There is an almost linear drift in the time series and it has an inital offset of around $7.6\,\mathrm{mm}$. Figure 4.12 shows an equally long excerpt from Bridge B. It contains 131 events with a mean duration of $1.75\,\mathrm{s}$. The offset is even higher with $16.38\,\mathrm{mm}$ due to disturbances from vehicles beneath the bridge. These disturbances can also appear in shape of high-frequency noise but without causing phase jumps. These are easily recognizable by their high deflections in the positive direction compared to the average. One considerable example for a disturbance can bee seen at approximately $670\,\mathrm{s}$. The long-term drift is less linear but more variable.

4.6.2 Methodology

Our methodology consists of two approaches. As a baseline, we apply threshold-based event detection and we compare it to the CNN from Section 4.5. Figure 4.13

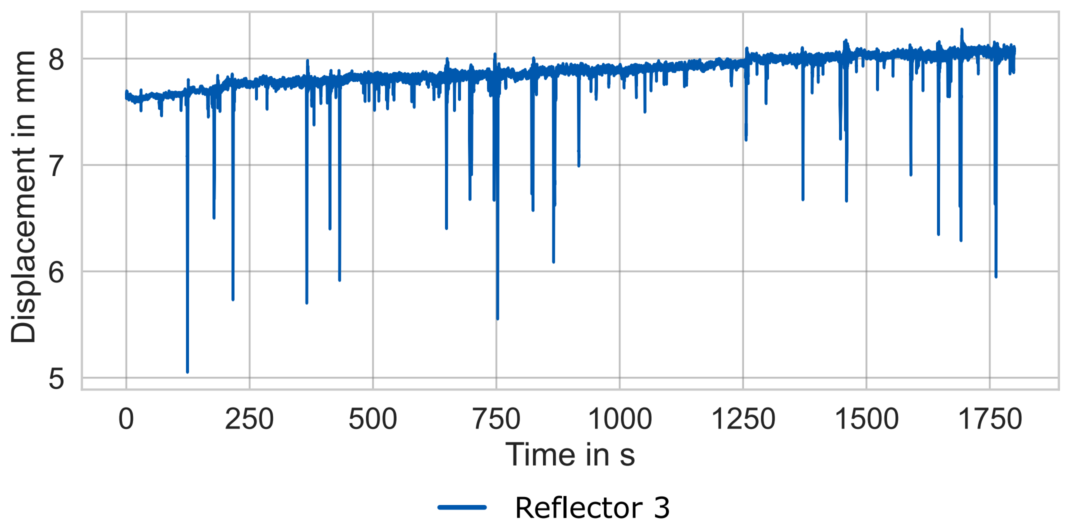


Fig. 4.11.: Time series signal of reflector 3 from Bridge A used for analyzing our approaches capability to detect events as a whole.

shows the methodology for the baseline. For comparison, we also use $0.5\,\mathrm{s}$ windows for event detection. The threshold is calculated based on the last 5 windows, which are not part of an event, according to:

$$\text{threshold} = \text{mean}(fifo) + 3 \cdot \text{RMSDMV}(fifo), \tag{4.1}$$

where $fifo$ is the first-in-first-out buffer of 5 windows. We initialize it by using the first 5 windows. Futhermore, we initilize the flag, which indicates if an event is occuring as false. For the next sample the mean is determined and compared to the threshold. In case it is smaller, an event occurs. If the previous sample is no event, this is regarded as the event start and the index is saved and the next sample is investigated. In case that the mean is bigger than the threshold, no event occurs. If an event is running, this event has now ended and the end time is saved. Otherwise we append this sample to our $fifo$-buffer, discard the oldest sample and update our threshold. All hyperparameters, such as window size and threshold value, have been determined manually. They are the same for both bridges.

The DL CNN-approach classifies all samples without any preprocessing except the lowpass filtering. Consecutive samples, that are classified as events, are summarized to one single event. Finally, we apply some domain knowledge. We filter out events with a length of up to $0.5\,\mathrm{s}$, as they would involve very high vehicle speeds. This is done for both approaches.

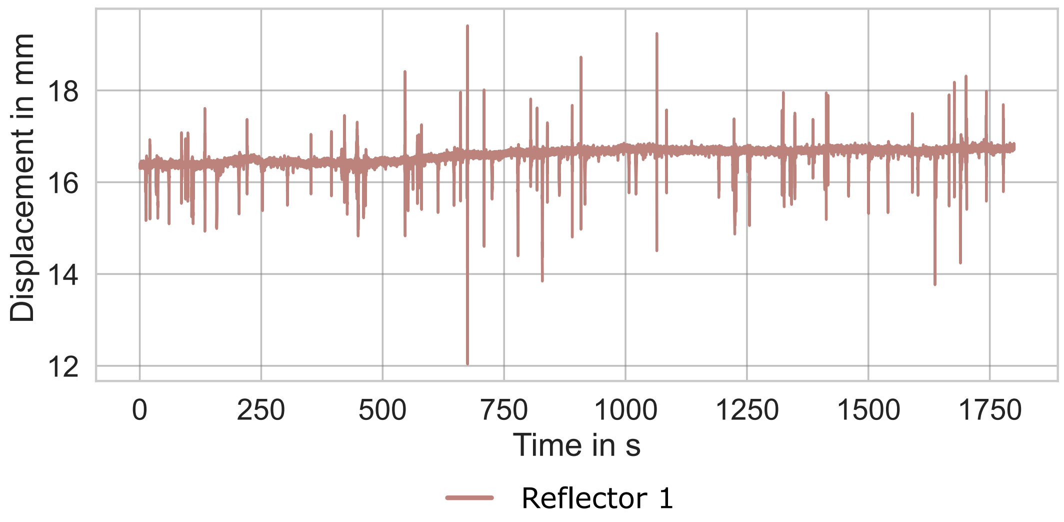


Fig. 4.12.: Time series signal of reflector 1 from Bridge B used for analyzing our approaches capability to detect events as a whole.

4.6.3 Results and Discussion

We start the evaluation of both approaches by examining the classification results, however we neglect correctly classified non-events. Then we investigate how closely start and end time are detected in the case of correctly identfied events. For that we use MAE and RMSE regarding the start time, and the end time. An event is regarded as detected correctly, if its reference event center lies within any detected start and end time area.

The detection results for Bridge A and Bridge B for both approaches are given in Figure 4.14 and Figure 4.15. Overall, it shows that CNN can detect events more succesfully. Especially at Bridge A only 3 events are missed, whereas 18 events are missed using a threshold. Conversely, the CNN mistakes more non-events as events. This is oftentimes during decay when high amplitudes are present. But also some non-events are misclassified although they show no visual resemblance to an event. Thresholding leads to only 4 wrongly detected events.

The same trends can be identified for Bridge B. Here, however, the CNN achieves worse results overall. One factor could be our dataset from Section 4.3, since it mainly consists of Bridge A data. And since the deflection of Bridge B is less pronounced, CNN could have difficulties. Another reason is the lower SNR due to our measurement setup. Finally, disturbances are mostly misclassified as an event by the CNN. Accordingly, the threshold approach has more problems identifying non-events. Regarding the misclassification of events, it achieves results comparable to Bridge A.

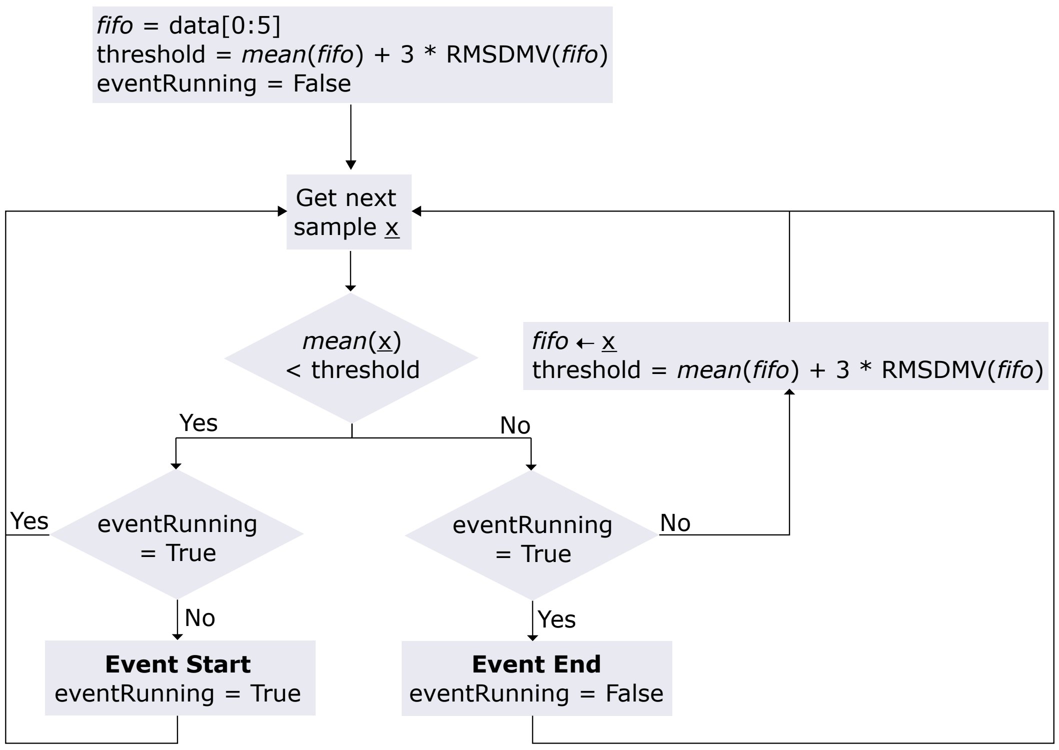
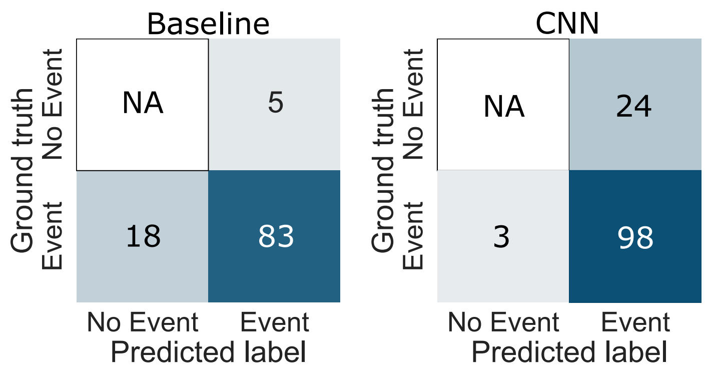


Fig. 4.13.: Methodology for our baseline approach using a dynamic threshold.



Baseline	Predicted: No Event	Predicted: Event
Ground truth: No Event	NA	5
Ground truth: Event	18	83

CNN	Predicted: No Event	Predicted: Event
Ground truth: No Event	NA	24
Ground truth: Event	3	98

Fig. 4.14.: Confusion matrixes of both approaches for Bridge A.

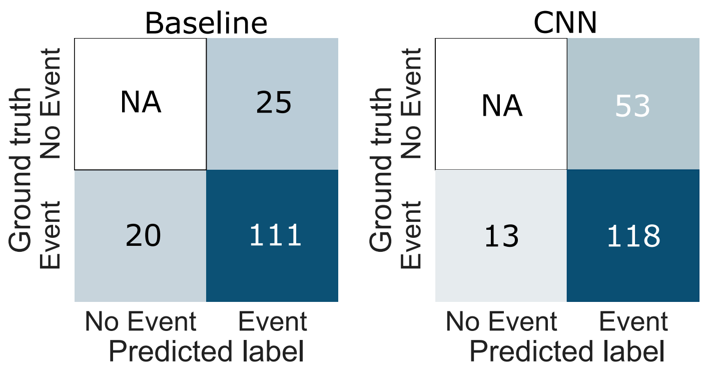


Fig. 4.15.: Confusion matrixes of both approaches for Bridge B.

One remaining challenge with our CNN is that if the window falls on the lowest point of an event it will probably mistake it for non-event, hence splitting the event in two. According to our definition of corretly detected events we would therfore miss such events although the slopes are detected correctly. Solutions might be a sliding window in general or within the two slopes. Yet, when several cars cross the bridge simultaneously, the displacement signal actually stays low for a longer time. There is no way for the CNN to know that windows within this time span are part of an event without context information. Thus, more constraints for edge cases are required. In the end, ad-hoc manual checking of such cases might be the easiest way to go with the CNN as only few of them occur.

On basis of correctly extracted events, we now investigate how well both approaches cover an event. For each detected event, we use the corresponding reference values for evaluation. To be more precise, we iterate over detected events and compare its timing with the start time of the first included event and the end time of the last one. For example in Figure 4.16, we compare the start time of the shaded area in the CNN plot with the first shaded area in the reference plot and the end time of the shaded CNN area with the end time of the second shaded reference area. We do this since the the CNN event contains both these events. The threshold-based approach only detects the first of these events, thus its start and end time are compared to the first shaded reference area only.

Table 4.6 states the evaluation results based on start and end time. As we only used correctly identified events, the data basis for the threshold-based and the CNN approach are slightly different. In general, differences between prediction and reference are to be expected, since we use $0.5\,\mathrm{s}$ windows, whereas the reference data has a higher temporal resolution. Furthermore, we labeled an $0.5\,\mathrm{s}$ window as event, when at least $0.2\,\mathrm{s}$ of it are contained within an event (see Section 4.4.1). Thus,

Tab. 4.6.: Results of the models, threshold-based and CNN, for Bridge A and Bridge B on how well they can extract whole events. They are evaluated with the MAE and RMSE on the detected the start time and end time.

Parameter	Model	Bridge A		Bridge B	
		MAE in s	RMSE in s	MAE in s	RMSE in s
Start time	Threshold	0.14	0.17	0.26	0.50
	CNN	0.22	0.35	0.14	0.20
End time	Threshold	0.20	0.35	0.25	0.37
	CNN	0.34	0.49	0.36	0.58

missing approximately the first or last $0.2\,\mathrm{s}$ of an event but also including $0.3\,\mathrm{s}$ more at the beginning and end are errors already inherent in the labels for the CNN.

Table 4.6 shows that the threshold method achieves superior results except for Bridge B start time. Overall, the MAE results between CNN and threshold for both bridges and parameters differ by approximately $0.1\,\mathrm{s}$. Thus, the results are close together. For CNN they lie within what we would expect from the reference labels. The RMSE is always close to the MAE indicating that not many outliers are present in the predictions. The threshold approach performs worse for Bridge B due to the lower SNR and thus the higher RMSDMV of the displacement signal. No such clear statement can be made about CNN when comparing both bridges. The start time MAE improves by $0.1\,\mathrm{s}$, yet, the end time MAE declines by $0.02\,\mathrm{s}$. The SNR seems to have a smaller impact on start and end time detection compared to thresholding.

A disadvantage of both RMSE and MAE is that the sign of the error is disregarded. However, for a subsequent analysis of extracted events it might be better to overshoot the event boundaries by a small amount than to neglect parts of an event and in turn maybe lose relevant information. Therefore, we further investigate the type of errors done by both approaches. Figure 4.16 depicts an excerpt from Figure 4.11 with the predictions of both approaches as well as the reference labels. Concerning the classification results, it documents well what we have described earlier. The threshold-based approach misses a multi event starting at $1660\,\mathrm{s}$. CNN detects all events, however, it misclassifies two non-events as events. Since the CNN misses the non-event window between the two events at around $1660\,\mathrm{s}$, these events are cumulated to one event. Overall, the CNN identifies the edges of events satisfactorily in this example. The only exception is a type of outlier which can be seen after the two crossings at $1068\,\mathrm{s}$. The CNN misclassifies some non-events after the second crossing. As explained before, at Bridge A heavy trucks can cause measurable displacement to the monitored field despite driving on the other field. This is

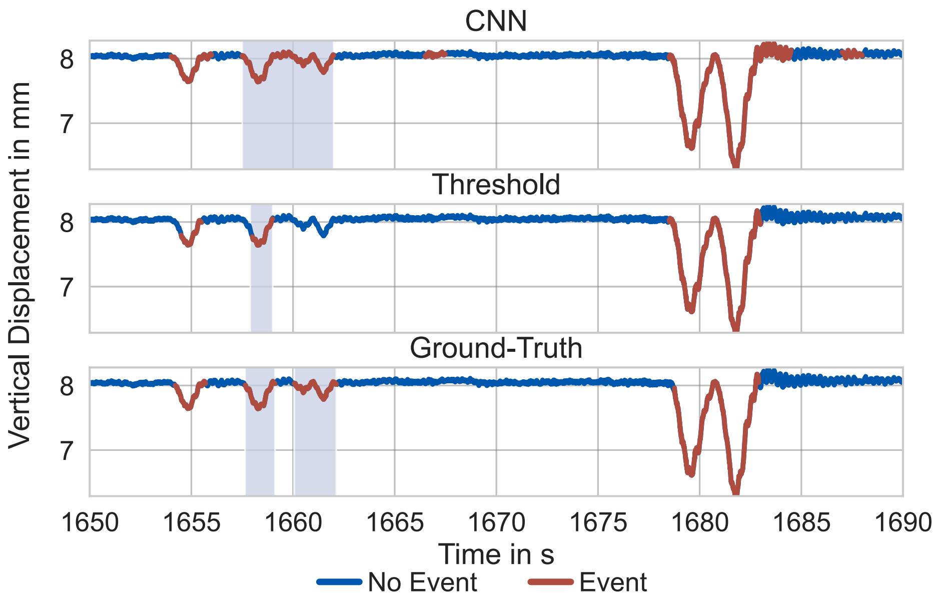


Fig. 4.16.: Example time series from Dietersdorf with the predictions from both approaches as well as the reference labels.

the case here. Thus, the CNN correctly detects a discaplement, however, it is not in accordance with our definition of an event. Therefore it counts as an error. Coincidentally, there are methods of SHM which use the curvature as it is extracted by the CNN for SHM [121, 38]. This does not happen for Bridge B, as there is only one field.

While our threshold-based approach does not show this behaviour, it oftentimes underestimates the duration of an event. Examples of this behaviour can be seen in the first two events, where the start point is detected too late, thus missing a small proportion of the overall event. Since the error is small, however, MAE and RMSE are small too. To highlight how a threshold-based approach underestimates the event duration, Figure 4.17 displays the Bridge B prediction error for all detected events colored according to the approach. We calculate the error by subtracting the predicted value from the reference. Therefore, for the start time, we want a positive value and for the end time a negative one. If that is the case no part of an event would be missed. The corresponding desired areas are shaded. As it can be seen, the CNN achieves better results in this regard. The threshold errors are more often of a type with which event information would be lost.

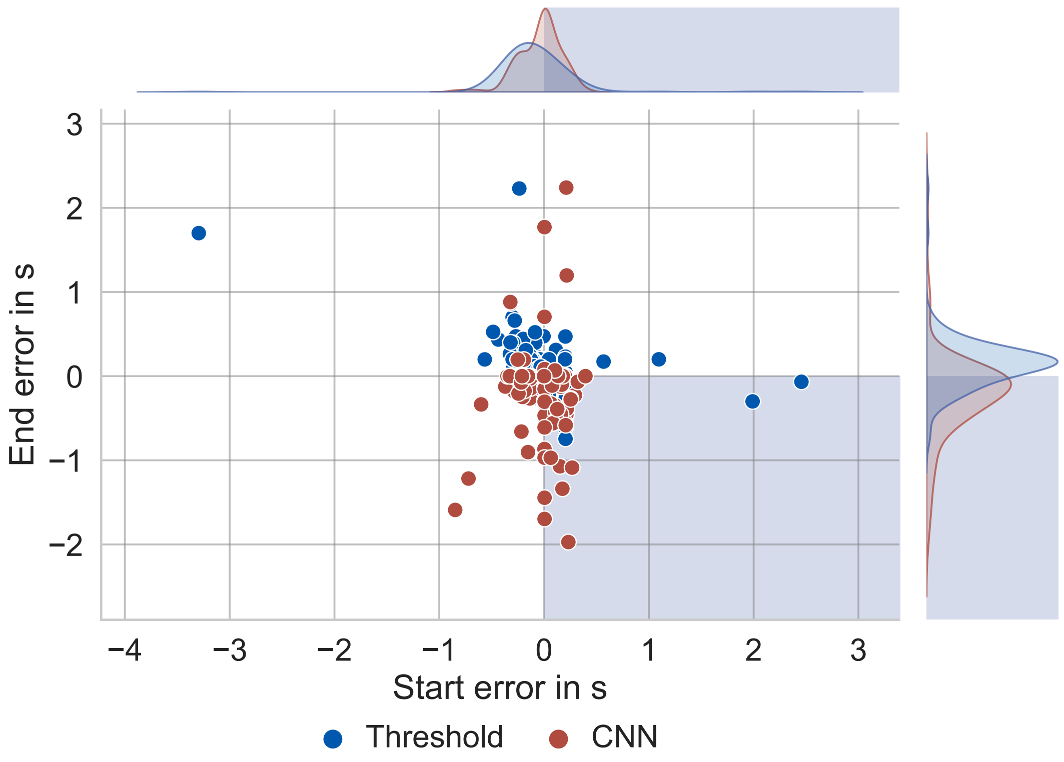


Fig. 4.17.: Distribution of both start time and end time error at Bridge B for both approaches. In the highlighted area no portion of an event is missed.

4.6.4 Conclusion and Outlook

In this section, we investigated the potential of DL to succesfully detect events from start to finish. As a baseline, we implement a simple threshold-based approach. Overall, our DL solution is superior to the threshold-approach. It detects events more succesfully and loses less information. Yet, we could also highlight some remaining challenges. For one, the CNN often misclassifies windows without events, especially for Bridge B. To tackle this, stronger filtering might be an option in order to reduce the SNR. Also more constraints can be introduced during a postprocessing step to filter, for example, high-frequency disturbances. This, however, requires more domain knowledge and jeopardizes transferability.

The CNN still detects events too late or predicts their end too early, albeit, less often than the threshold approach. This could lead to difficulties with follow-up algorithms, which build upon correctly extracted events. It might be worth to revise our event definition and how events are labeled. Another approach would be to broaden the receptive field of the CNN so that it gains more contextual information. During training and testing of new models, special attention should be paid to edge cases. Until the CNN is more robust, a combination of CNN and thresholds could also be considered. Overall, we could show that our approach can succesfully extract events from beginning to end.

4.7 Conclusion

In this chapter, we developed a new data-driven approach to detect and extract bridge crossing events. We investigated shallow learners on two different datasets and a CNN DL approach on the more extensive dataset. Since we wanted to acquire the whole time series of an event, we finally evaluated the capability of our CNN approach to correctly identify an event from start to finish. As a baseline contender, we implemented a simple threshold-based approach.

For our shallow approaches, we tested eight models on only manually crafted features with different levels of preprocessing. It showed that with our manually crafted features, event detection is possible but pure bridge oscillation is generally misclassified as an event. Exploiting end-to-end DL via CNN improved our predictions in this regard. Despite some remaining challenges, CNN could extract an event from start to finish. With this achievement, we fulfilled our **Research Goal I**. The results of this extraction can be used to feed algorithms focusing on obtaining vehicle information with appropriate input data.

Part II

Event Classification

5 Vehicle Count

At this moment an unending stream of traffic was just going over the bridge.

— Franz Kafka
(Author of The Judgment)

This chapter includes material from the following works:

Matthias Arnold and Sina Keller. "Machine-learning for analyzing bridge displacement using radar data". In: *Bridge Maintenance, Safety, Management, Digitalization and Sustainability*. Vol. 1. CRC Press, June 2024, pp. 1405–1412

It is cited as [9] and marked with a brown line.

M. Arnold and S. Keller. "Machine Learning Approaches for Vehicle Counting on Bridges Based on Global Ground-Based Radar Data". In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* X-2-2024 (2024), pp. 1–8

It is cited as [6] and marked with a grey line.

Matthias Arnold and Sina Keller. "Machine Learning and Signal Processing for Bridge Traffic Classification with Radar Displacement Time-Series Data". en. In: *Infrastructures* 9.3 (Mar. 2024). Number: 3 Publisher: Multidisciplinary Digital Publishing Institute, p. 37

It is cited as [8] and marked with a green line.

5.1 Introduction

Acquiring information about vehicles on top of a bridge is the overall goal of Part II - Vehicle Classification. As described in the previous Chapter 4, our event detection approach identifies the presence of vehicles and retrieves these sequences. Other than that, no insight into the traffic, like the number of vehicles involved, is gained. Say, two trucks are crossing the bridge at the same time, this is treated as one event. However, many SHM and BWIM methods focus on single events, e.g. events with only one vehicle on top of the bridge [114, 64]. Identifying single events is thus necessary in order to apply them correctly. Even more so, algorithms for vehicle parameter extraction can be deployed in a more targeted manner if knowledge of the exact vehicle count is available [105].
The objective of this chapter is therefore to obtain that very knowledge from variable-length multi-variate displacement time series. To this end, we apply data-driven ML methods for classification and regression. The unequal duration of crossing sequences poses a challenge in this respect as input of variable size is rarely studied in the context of ML. Furthermore, we only have a small and skewed dataset on which we can develop our methods. Again, we aim to reduce the amount of preprocessing to a minimum to achieve transferability.

In Section 5.2, we will first describe the related work focusing on variable-length time series ML, quantity estimation in different fields and vehicle counting in bridge monitoring. Then, the dataset used in this chapter is specified in Section 5.3. Afterwards, we will present two different studies on vehicle count estimation. Figure 5.1 lists two objectives as well as the methodological approach embedded in the overall structure of this work. First, we will try to simply classify events depending on whether only one (single event) or several (multi event) vehicles are present during an event (see Section 5.4). This distinction allows us to focus on single events in the following chapters. For a more detailed traffic analysis, however, multi events should not be discarded. Therefore, we also try to estimate the exact number of vehicles involved in an event by means of classification and regression in Section 5.5. Both studies exploit shallow learners and harness data augmentation to deal with the small and skewed dataset. We also thoroughly survey data augmentation effects, feature importances and transferability exploiting SHAP. Finally, we review the applicability and transferability of data-driven event classification using ML in Section 5.6.

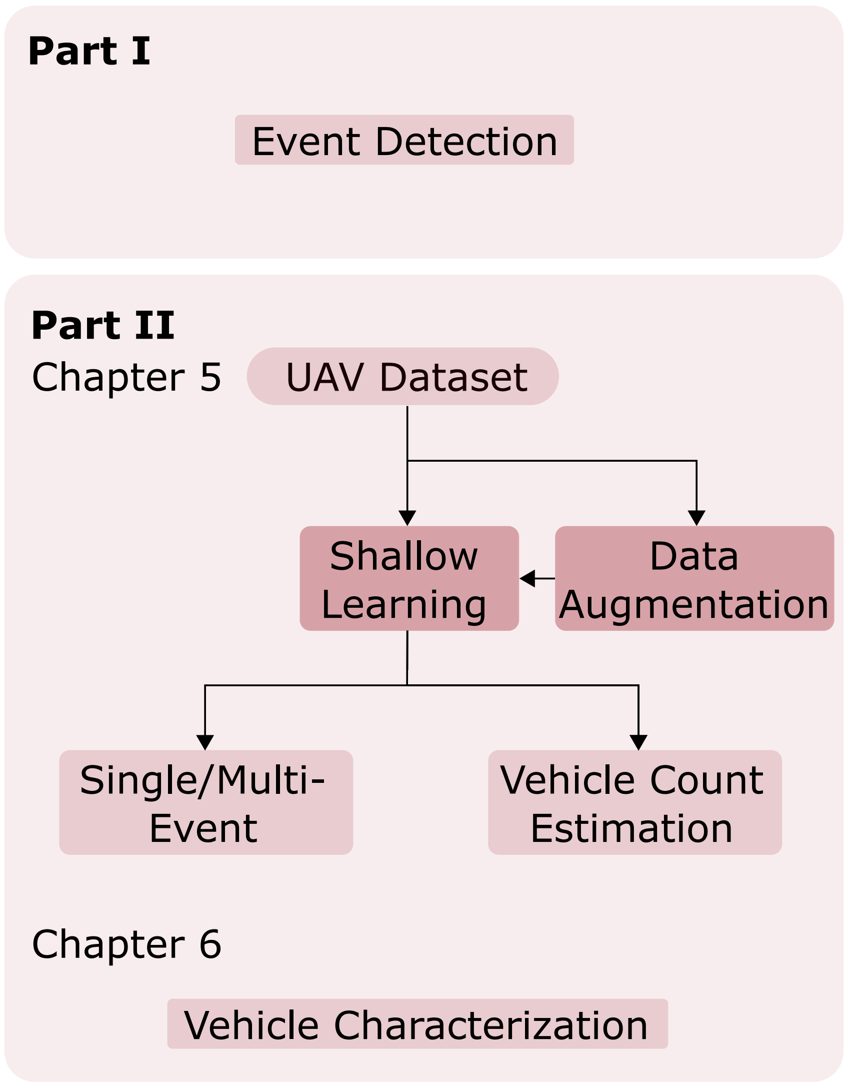


Fig. 5.1.: Overview of the vehicle count estimation chapter.

5.2 Related Work

Following the state-of-the-art time series classification approaches detailed in Section 4.2, we will now place our focus on variable-length time series tasks. Thereupon, relevant work concerning count estimation in different fields is described. To conclude, we will present studies in the context of bridge monitoring which discuss the vehicle count.

The relevance of time series data in ML has never measured up to the prominence of images, although the variety in the data poses an exciting challenge. A wide range in scaling, length, sampling frequency, channels, etc., makes it challenging to develop versatile models to handle such complexity. The UCR dataset provides extensive datasets and is often used as a benchmark for new approaches [30]. However, at the date of writing this thesis, only six datasets are both multi-variate and of unequal length. Moreover, the differences in signal length are minor. Oftentimes, an unequal length is circumvented by rescaling or padding in the time domain or extracting features, for instance, by fitting an autoregressive integrated moving average model [17]. Doing so allows the usage of established ML methods. In case of rescaling, information can be lost if sequences are resampled. Padding may influence the prediction as the model can learn from it. Hertel et al. [60] took an approach based on CNN in combination with padding and masking for acoustic scene classification. To this end, they implemented a global pooling layer that supports masking to prevent the model from learning the padding. Masking forces a layer to ignore masked values. They do not go into detail regarding their implementation as the build-in convolutional layer of tensorflow does not support masking [129]. Schneider et al. [136] implement a similar network based on masking in the industrial context to discern good and bad plastic products. They use tensile force time series as input which can vary heavily in amplitude and duration. For comparison, they exploit DTW which can handle input of unequal size. Their CNN achieves slightly better results than the DTW while being considerably faster. Their shared code, however, reveals that they use the build-in convolutional layers. Thus, despite their intentions, they do not apply masking succesfully and the layer can learn from the padding.

The effect of padding on model peformances is to this day rarely taken into account, let alone studied explicitly. Dwarampudi and Reddy [37] investigate the impact of pre- and post-padding, namely if the time series is padded before the start or at the end, on LSTM and CNN. They claim that it does effect the LSTM as they retain a memory whereas the CNN remains unaffected. Then again, LSTM can handle masking, thus, padding should not be a problem. A more exhaustive and systematic study has been conducted by Rio et al. [129] in the context of archaeal protein functional prediction. They evaluate several padding methods and come to the conclusion that padding does indeed influence model performance. Unfortunately, a comparison of effects on different architectures has not been within the scope of their study. Overall, padding for DL needs to be treated with care. As mentioned, RNN like LSTM can be exploited in combination with padding [71, 72, 113]. Yet, for small datasets, LSTM tend to overfit. Training the network on one instance at a time, e.g. using a batch size of one, offers an alternative to padding in DL. This comes with potential major sideffects like longer training time, and lower accuracy due to missing batch normalization [147].

Another approach is to project all sequences into a feature space with fixed size.

We describe the most famous feature extraction methods in Section 4.2. To this date, most of these models are not implemented in a way in which variable-length data can be inputted, emphasizing again the low status of this subfield in ML. One exception is MiniRocket. Its developers have contributed a version of MiniRocket to the Python library sktime [90] which can handle unequally long time series. We provide more details about MiniRocket in Section 3.2.2.

Detecting and segmenting objects in images has long been a focus of ML. Fang et al. [40] raise the question –and answer it positively– if RNN can even count objects in images. The objective of counting incidents within a time series also exists although not to this extent. In human activity recognition, for example, step counting is one of the most predominant applications in current days due to the availability of smartphones and -watches [12]. Khan and Abedi [77] introduce a LSTM-based approach using smartphone accelerometer data. They test it on several public datasets with time series data of variable length.
Another application requires counting specific actions in order to monitor workout routines. Bian et al. [15] explore capacitive sensors to classify and count repetitions in seven different workout activites. For counting, they use a threshold-based approach after filtering the time series signal. The filters are adapted to the specific routine, requiring manual labor. Sopidis et al. [144] present a LSTM model which counts activities from weakly labeled acceleration data. They generate variable-length sequences by concatenating different activities. The model is trained to output the number of each activity as a regression task after witnessing the whole sequence. Concurrency of different movements is not investigated. Liu et al. [86] count human presences by leveraging the impact of human activites on WiFi signals. These signals are transmitted and received by a standard MIMO WiFi device. After preprocessing, a ANN is used to determine the presence of up to five persons. The measuring period is constant but variety in activity and position is examined. For the duration of each measurement the number of people is fixed.

Many approaches for vehicle counting systems also apply ML, but they use images from surveillance systems as input data [39, 54]. However, visual techniques require an unobstructed view and might fail, e.g., in the event of snowfall or fog. Therefore, other methods are being researched. Taghvaeeyan and Rajamani [145] exploit four magnet sensors at the roadside for vehicle counting and classification. Passing vehicles induce peaks in the measured time series data and with a simple threshold 186 of 188 vehicles could be detected. They embed their study in the context of traffic surveillance and not bridge monitoring. Thus, for BWIM further sensors are required.
BWIM studies oftentimes neglect the aspect of vehicle counting or the classification

of single and multi vehicle crossings. Presumably, many studies using acceleration or strain data consider vehicle counting a straightforward task similar to detecting events, as detailed in Section 4.2. As both signals capture local features of the bridge, axles can be detected more easily. The only study using displacement data for BWIM, revert to artificially generated single events as well as computer vision [115]. Yet, many standard BWIM SHM methods cannot handle multiple-presence crossings and instead require single-vehicle events [114, 64]. Moghadam et al. [105] introduce a method for calculating a bridge influence line from multi-presence crossings. However, they recommend using a different approach for single crossings since it is more exact, making a differentiation between single and multi events desirable. Kawakatsu et al. [76] use DL for BWIM. They train CNN to classify traffic using equally long strain time series. According to their study, the CNNs can handle multi-presence events. However, they do not study the effects of several vehicles.

Research Gap In summary, a gap exists concerning the distinction of single- and multi-presence bridge crossing events based on data-driven displacement time series classification. Moreover, extracting the vehicle count from events has not been studied to this day. A major challenge hereby lies within the variable-length of the time series data. MiniRocket has proven to be a performant approach in such situations. Therefore, we investigate MiniRocket and shallow learners for single versus multi event classification as well as vehicle count extraction.

5.3 Dataset

The UAV dataset used in the studies presented in this section are based upon the UAV measurements introduced in Chapter 2. It contains GBR measurements for which UAV reference data is available. Table 5.1 gives a general overview of the dataset for both bridges. As mentioned before, disturbances occur at Bridge B. We will only regard undisturbed events in this work.

Tab. 5.1.: Overview of the UAV-recorded events used for the event classification. Adopted from [9].

Bridge	Vehicles	Trucks	Cars	Events	Single Events	Multi Events
A	1110	341	769	962	849	112
B	491	114	377	343	271	72

Tab. 5.2.: Overview of vehicle counts in events. Adapted from [6]

Bridge	1 vehicle	2 vehicles	3 vehicles	4+ vehicles
A	849	84	24	5
B	271	35	22	15

Overall, 1110 vehicles have been recorded for Bridge A and 491 for Bridge B. Since an event can contain several vehicles, the number of events is smaller, with 962 events for Bridge A and 343 for Bridge B, respectively. Besides the small dataset size, the imbalance of the dataset poses another challenge. This is true in two ways: Firstly, multi events are scarce compared to single events. Secondly, trucks appear less often than cars. Since trucks are inherently slower than cars and cars tend to cue behind a slow truck, trucks are underrepresented in single events. In the case of Bridge A, for example, 241 trucks are single events, whereas 608 single events contain cars.
Table 5.2 gives a more detailed overview dividing multi events according to the number of vehicles involved. As one vehicle events represent single events, they make up the vast majority. The highest number of vehicles during one event amounts to 5 for Bridge A and 7 for Bridge B, respectively. However, events with more than five vehicles rarely occur; therefore, we cluster these events in a 4+ class.

While single events can differ significantly, multi-presence events come in an even greater variety: For instance, two heavy trucks might cross the bridge simultaneously in opposite directions. This scenario would be very relevant for SHM due to the significant stress induced on the bridge. Still, it's also challenging to visually differentiate from a single event using only time series displacement data. To better understand, Figure 5.2 shows one single and one multi event for each bridge, although these examples do not cover the broad spectrum of traffic situations in the dataset. The corresponding time series data for all reflectors is drawn below each UAV image. For each vehicle, its time on the monitored field is highlighted. Since there are overlapping areas in the lower plots, these are classified as multi events. Bridge A multi event shows a truck finishing its overtaking maneuver of another truck on the bridge field. The time series already resembles a single event of a very slow vehicle, e.g., when compared to the Bridge B single event. Thus, parallel crossings are barely distinguishable from single events. Attributing the maximum displacement or the maximum load of such an event falsely to one vehicle would lead to overestimating the bridge load and possibly to unnecessary reinforcements. A small car follows after the two trucks, but the vibration heavily masks the bridge bending the car causes. Finally, the Bridge B multi event shows a more straightforward example of two consecutive cars.

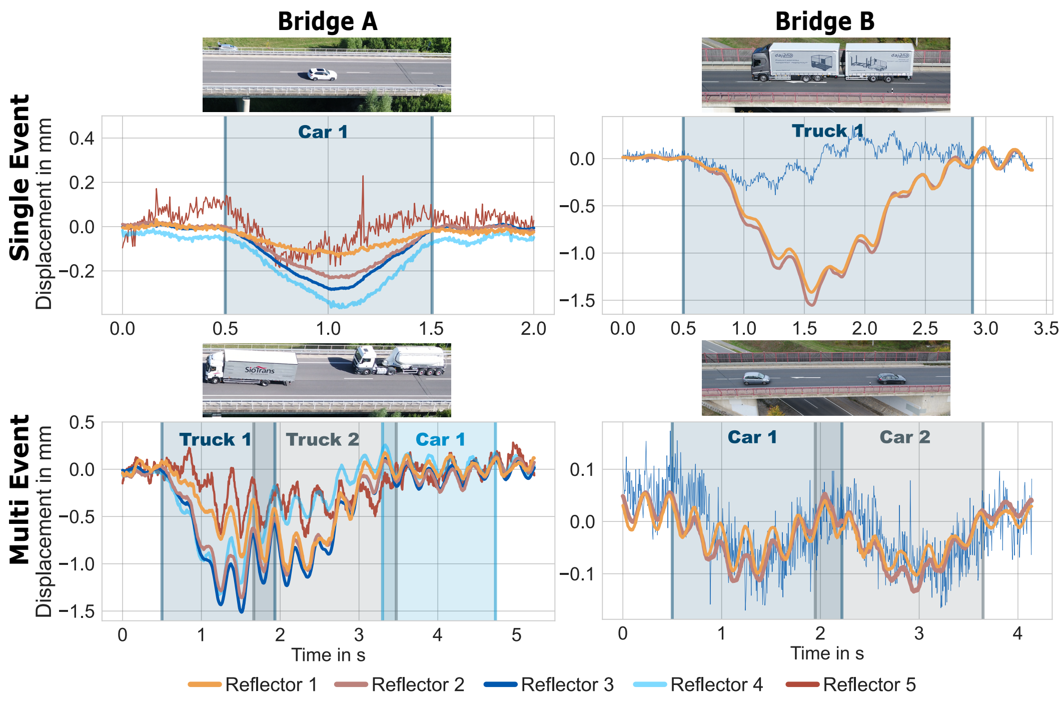


Fig. 5.2.: Example for a single (upper) and a multi event (lower) for Bridge A and Bridge B. Adapted from [9] by changing the color palette.

Figure 5.3 depicts two more multi events, focusing on two whose displacement signal resembles single event curvatures. The Bridge A event in Figure 5.3 depicts two cars and a truck. The first car overtakes the truck. The second car drives in the direction opposite to them. Visually, this event could pass as a single event.

For a comprehensive anlaysis of traffic situations, it would be beneficial to know the number of vehicles involved in a multi event. Even more so if the vehicle count would be split into the number of trucks and cars. As cars usually have two axles, this information can be helpful to determine the truck properties more accurately. The example of Bridge B visualizes two trucks driving in opposite directions (see Figure 5.3). Although it can be recognized as a multi event, the number of vehicles is hard to percieve by humans.

Figure 5.2 gives the impression that it could be sufficient to monitor the length of an event or its maximum displacement for one or two reflectors to successfully classify it. To illustrate that the distributions for both classes overlap significantly regarding such features, Figure 5.4 shows the duration of an event and the maximum displacement for reflector 3 of Bridge B and their corresponding Kernel Density Estimation (KDE). Although multi events tend to take longer, two consecutive cars are often faster than a single truck. And events of two trucks crossing the bridge in parallel on opposite lanes might last as long as a single truck event. On the other hand, most

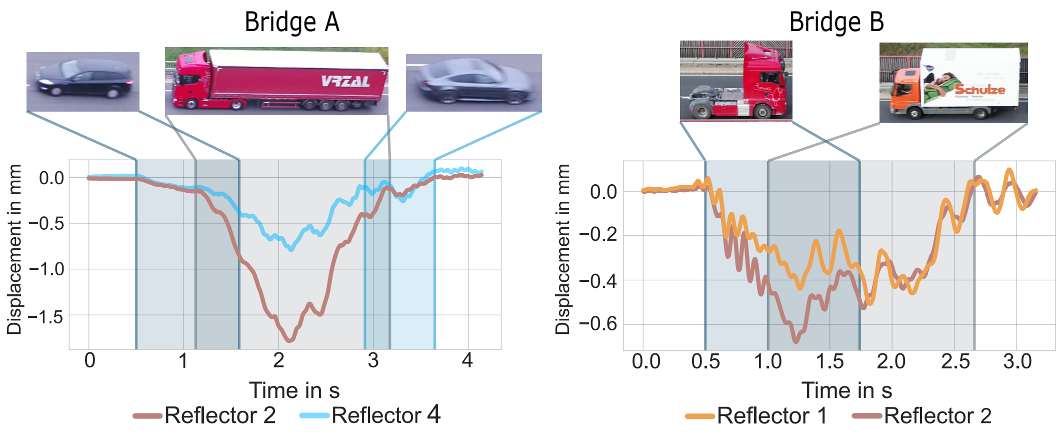


Fig. 5.3.: For each bridge, a multi event is shown. In the time series plot, the respective time of each vehicle on the monitored field is highlighted and attributed to the corresponding vehicle images. Adapted from [6] by changing the color palette.

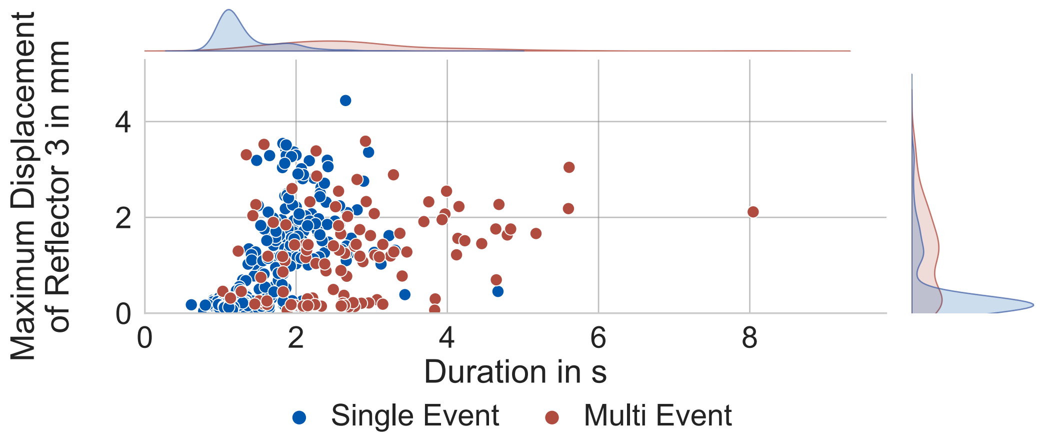


Fig. 5.4.: Scatterplot of maximum displacement and duration with normalized KDE. Adapted from [9] by changing the color palette.

multi events have a higher deflection since slow and heavy trucks are involved, yet a clear decision boundary is not apparent. Although these two features alone are not enough to make a correct classification, additional features in combination with ML might be more successful.

A likewise conclusion can be drawn for Bridge B. Figure 5.5 depicts the distribution of event duration grouped into single and multi events for both bridges. Compared to Bridge A, Bridge B is even more skewed towards a longer duration for multi events. Thus, the duration can indicate the number of vehicles on the bridge. Yet, both distributions overlap, so additional features need to be found. Furthermore, Figure 5.5 highlights the dataset imbalance, as there are many more single vehicle events than multi events.

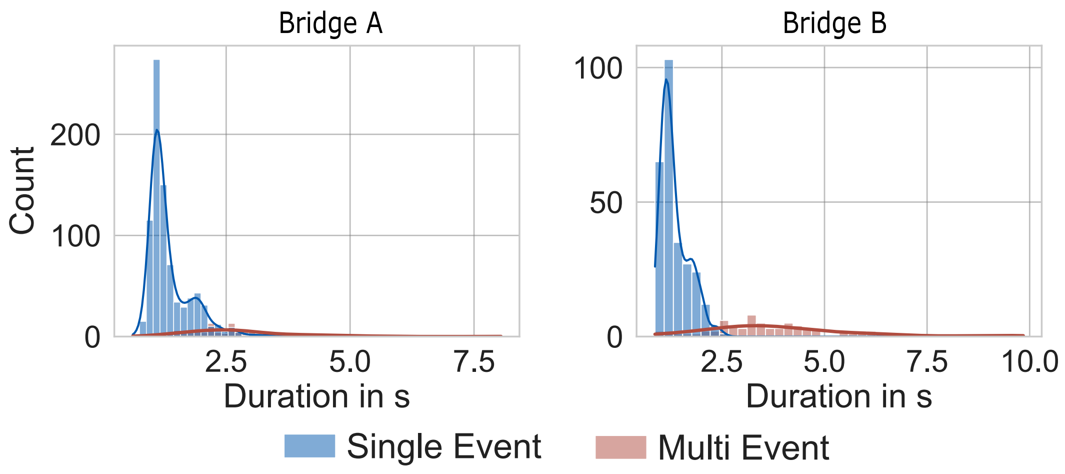


Fig. 5.5.: Distribution of event duration for both bridges grouped by single and multi events. A more detailed breakdown of multi event into vehicle count classes has been omitted for presentation reasons. Adapted from [9] by changing the color palette.

5.4 Single Event Detection

In this section, we will focus on classifying events in single events where only one vehicle is present on the bridge and multi events with several vehicles. As a first step, we use Bridge A data for training and testing and Bridge B data for testing only in order to investigate transferability to completely unknown structures. For comparison, we also train and test on both datasets combined. Finally, we will survey the effects of data augmentation on a RF.

5.4.1 Methodology

In this chapter, we will shortly explain the methodology of the single- and multi-presence classification. Our proposed methodology consists of three steps: the data augmentation, the feature extraction and the ML models to classify the events. Figure 5.6 provides the schema of all steps.

Preprocessing

As evident from Section 2.2, bridges can have a varying amount of measurement points. For this study, we assume that at least two points are available at each bridge.

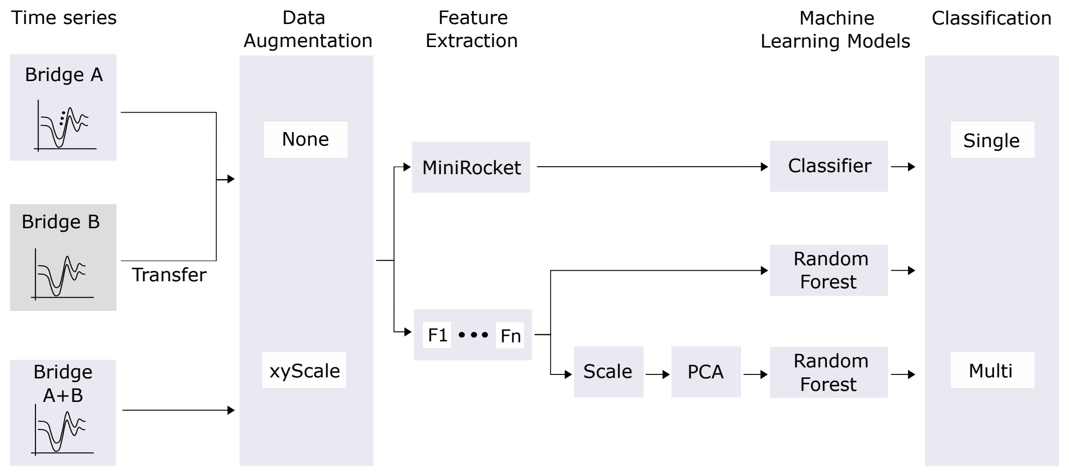


Fig. 5.6.: Schema of the methodological classification approach for single event classification. Adapted from [9] by adding Bridge A+B as data source.

Therefore, we use the two reflectors 2 and 4 for Bridge A and 1 and 3 for Bridge B, respectively. By this selection, we try to acquire information about the driving side, while retaining a high SNR. For comparison, we also train our models with all five reflectors of Bridge A. We split Bridge A data into training, validation, and test sets using a $70 : 15 : 15$ ratio, with each subset mirroring the overall distribution of single and multi events. In order to investigate transferability, we did not train our models on the dataset of Bridge B. Then, we combine both bridges and split the merged dataset again in a $70 : 15 : 15$ ratio.

In accordance with Ruiz et al. [133], we try to minimize the amount of preprocessing, i.e., we did not normalize the time series. We ensure that all time series start at a displacement of $0\,\text{mm}$ by removing the offset. As described in Section 5.3, single events tend to have a smaller maximum deflection. To mitigate this bias, we apply data augmentation in the form of y-scaling. We choose five values in the range of $-0.1\,\text{mm}$ to $-4.0\,\text{mm}$ (see Figure 5.4) superimposed with random values, in which the maximum displacement of each event would be scaled. To tackle the bias in duration, we oversampled all single events by factors 2 and 3 and downsampled all multi events accordingly. The newly created values extend our inital data set. We combine both scalings (`xyScale`) and compare it to no augmentation (`None`).

Feature Extraction

All our models are using features extracted from the time series. The manually selected features of the RF classifiers are listed in Table 5.3 and explained in Section 3.1. We have extracted every feature for each used reflector time series. With

Tab. 5.3.: These 14 features have been extracted from the GBR time series data for each reflector. Information about how these features are calculated can be found in Section 3.1. Adapted from [9].

Feature No.	Name of Feature
1	Maximum
2	Minimum
3	Mean
4	**rmsdmv! (rmsdmv!)**
5	Skewness
6	Kurtosis
7	Median
8	Length
9	Quantile25
10	Quantile75
11	NbrPeaks
12	xMinPosRatio
13	Power
14	MAD

14 features, this yields 28 features for two reflectors and 70 for five reflectors, respectively. We also train an RF after scaling the features and applying PCA to reduce the dimensionality. We use eight components since they explain over 95 % of the variance for the `None` dataset. Furthermore, we use MiniRocket for automatic feature extraction, as explained in Section 3.2.2.

Machine Learning Models

Using the validation set, we apply grid search to the models in the second to last column in Figure 5.6 to find the optimal set of hyperparameters. Since the dataset is imbalanced, we use BA as the score for the grid search. Furthermore, we apply class weights during training. The features extracted from MiniRocket are passed to a classifier. For small datasets with less than 10 000 observations, like in the `None` case, a ridge classifier is recommended by Dempster et al. [35]. Otherwise, logistic regression, which we use for `xyScale`, is suggested. Since MiniRocket is independent of y-scaling, we might introduce overfitting to the classifier with our data augmentation approach.

Tab. 5.4.: Test results for all classificators for single vs. multi event classification. The highest BA for each dataset is highlighted. Adapted from [9] by adding results for both Bridge A and Bridge B.

Bridge	Number of Reflectors	Data Augmentation	Model	OA	P	RC	BA
A	2	None	RF	0.91	0.97	0.93	0.85
			PCA RF	0.94	0.95	0.98	0.82
			MiniRocket	0.95	0.97	0.98	0.87
		xyScale	RF	0.90	0.94	0.93	0.76
			PCA RF	0.90	0.95	0.94	0.79
			MiniRocket	0.98	0.98	0.97	**0.90**
	5	xyScale	RF	0.94	0.94	0.99	0.76
			PCA RF	0.95	0.97	0.98	**0.87**
			MiniRocket	0.94	0.96	0.97	0.84
B	2	None	RF	0.96	0.97	0.98	**0.94**
			PCA RF	0.92	0.97	0.93	0.90
			MiniRocket	0.95	0.97	0.96	0.93
		xyScale	RF	0.59	0.92	0.57	0.69
			PCA RF	0.84	0.89	0.92	0.74
			MiniRocket	0.59	0.97	0.50	0.72
A+B	2	None	RF	0.89	0.96	0.92	0.83
			PCA RF	0.89	0.95	0.93	0.80
			MiniRocket	0.96	0.96	0.99	**0.87**
		xyScale	RF	0.83	0.96	0.83	0.81
			PCA RF	0.78	0.96	0.77	0.80
			MiniRocket	0.95	0.96	0.98	**0.87**

5.4.2 Results and Discussion

The study's objective is to investigate the potential of ML approaches to classify GBR variable-length time series vehicle crossings. Table 5.4 shows the results of the single vs. multi event classification concerning OA, P, RC and BA.

Concerning Bridge A with two reflectors, all models have an OA, P, and RC above $90\,\%$. However, these values are heavily skewed since single events are much more present. Therefore, we included BA, the average of recall from each class. All BAs are close to each other, between $82\,\%$ to $87\,\%$, with MiniRocket having the highest score. These results are deceptive, as there are correlations between duration and maximum displacement and event type, as described in Section 5.3. Figure 5.7 highlights this aspect. The decision boundary for MiniRocket in the left plot almost exclusively depends on the duration, with every event longer than $2.5\,\mathrm{s}$ classified as

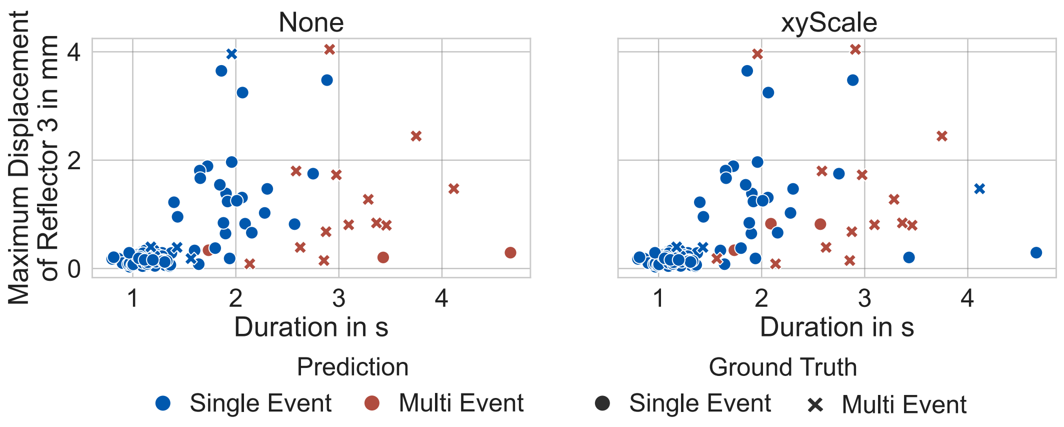


Fig. 5.7.: Test set classification results for MiniRocket with `None` and `xyScale` data augmentation. Adapted from [9] by changing the color palette.

a multi event. The boundary becomes more complex after applying `xyScale` data augmentation. Fewer long single events are now misclassified as multi events, as seen in Figure 5.7 on the right. Conversely, MiniRocket has more difficulties with medium-length single events. The effect of `xyScale` also shows in the BA of all three models. RF and PCA RF achieve a worse BA since they have to find more complex relationships. MiniRocket improves with data augmentation, achieving the best results for two reflectors on one bridge, with a BA of 90 %. Considering the small size of the dataset and without providing any prior domain knowledge, these are very auspicious results. Figure 5.7 also shows that short multi events are still misclassified as single events. Heavier augmentation in time might solve this issue. When using five reflectors, we did not expect any additional information in those time series, but on the contrary, to achieve worse results due to the higher complexity. Accordingly, RF and MiniRocket achieve equal or worse results. Interestingly enough, PCA RF has better results than with only two reflectors. One explanation could be that the driving side is more distinct with more data, which can help detect multi events.

For a more in-depth analysis regarding data augmentation effects and our developed features, we use SHAP on the RF (see Section 3.2.5). We focus on models trained on two reflectors. Our analysis is true to the model, e.g. feature dependencies are broken but out-of-distribution features might be passed to the model. Figure 5.8 shows the beeswarm plot generated by SHAP. A beeswarm plot is a very dense form to visualize feature importances and Shapley values. On the left side, features are ordered by their importance from top to bottom. Irrelevant features are summarized for clarity. We append the reflector number after each feature name. Thus, "Length0" describes the feature "Length" of the first input reflector and "Length1" the same feature of the second reflector. For every data point of the Bridge A test set the

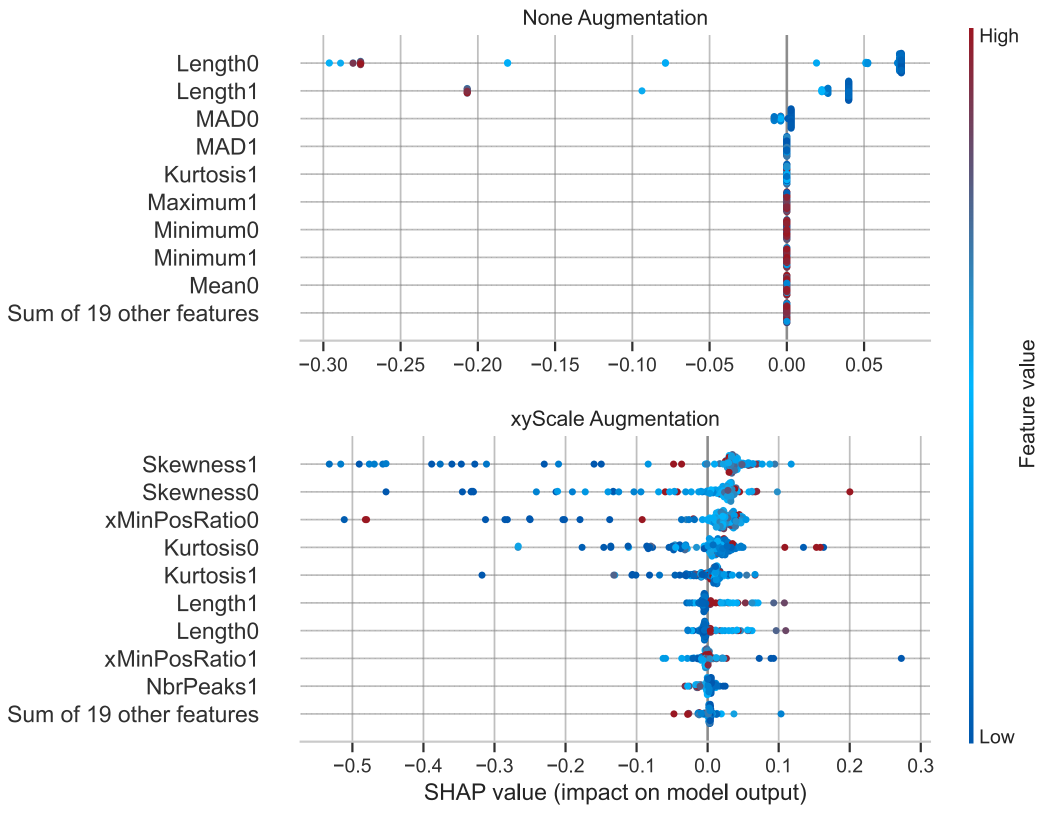


Fig. 5.8.: Beeswarm plots for `None` (top) and `xyScale` (bottom) data augmenation.

SHAP value for all features are calculated and represented by a scatter point on each feature line. If many scatter points fall on the same position of the x-axis, they get slightly distributed along the y-axis to show density. The SHAP value describes the amount a feature contributes towards predicting a data point as a single event. Lastly, all feature values are scaled to their own range and the scatter points are colored accordingly. For instance, the color red stands for a high feature value, e.g. a long signal, when looking at the "Length" feature.

When looking at the beeswarm plot for `None` augmentation at the top of Figure 5.8, it shows that both length features are the dominant features. This is expected as we indicated this dominance before. Focusing on "Length0", the color-coding reveals that a low length value, e.g. short events, push the prediction with a value of approximately 0.05 towards single event. Even more so, a high signal length heavily contributes to predicting multi event.
The `xyScale` augmentation in the plot below manifests that scaling in the temporal dimension helps to reduce the importance of the signal length feature. Now, other features are more important and the impact is distributed more evenly. "Length1" and "Length0" are now in position 6 and 7 regarding their importance with maximum

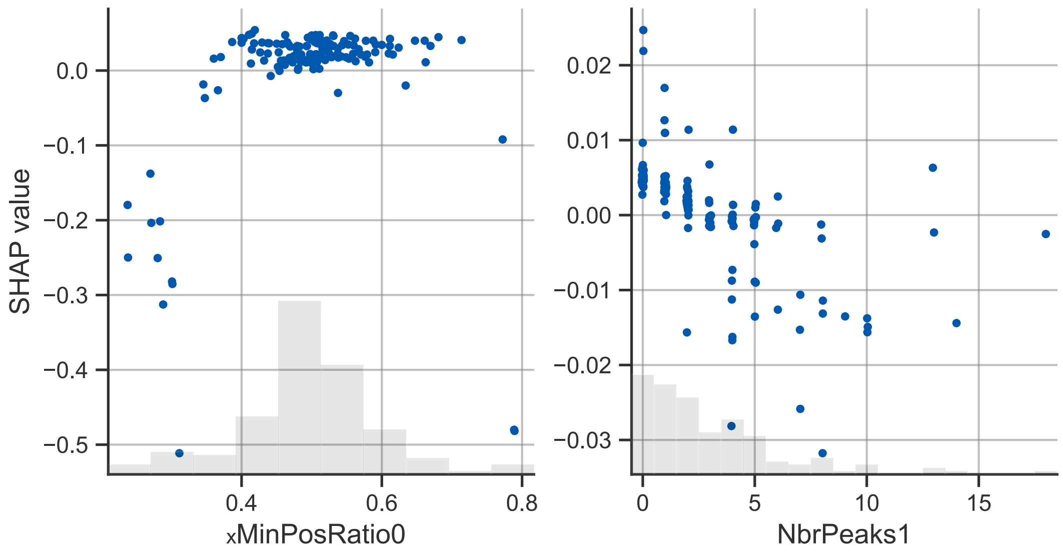


Fig. 5.9.: SHAP values for our self-developed features “xMinPosRatio” and “NbrPeaks”.

abolute impact of circa 0.1. Interestingly, a longer event duration is associated with single events. New front-runners are the skewness for both reflectors. They contain information about the lane, thus, it makes sense to include both. In either case, a high value leads to a higher single event probability. Our self-developed feature “xMinPosRatio” takes third place for the first reflector. The second manually crafted feature “NbrPeaks” is of lower importance on 9. place. Still, we will shortly investigate their impact to see if the model follows our intuition when designing them.

To this end, Figure 5.9 depicts the SHAP value over the feature value as a scatter plot. “xMinPosRatio0” is harnessed by the model as intended for Bridge A `xyScale`. Low and high values push the model to predict multi events, otherwise single events are promoted. “NbrPeaks1” has a less obvious correlation. As single events can be superimposed by oscillation, for instance when following a truck, the feature is less important. Still, the support of multi events increases with the number of peaks found. The hyperparameter tuning for “NbrPeaks” was done on data without augmentation. So, the RF could also be confused, as the temporal scaling via over- und downsampling might affect the number of detected peaks.

Finally, the tests on Bridge B show that a transfer to an unknown dataset can be successful to a certain degree. With `None` augmentation, all models suffer again under the correlation depicted in Figure 5.7. The BA is even $90\,\%$ and above. This score is partially due to the more extensive dataset, where usual events, e.g., cars following a truck, are more prominent, whereas complex events, such as two trucks simultaneously, are rare. Also a decision boundary for Bridge B on the basis of the

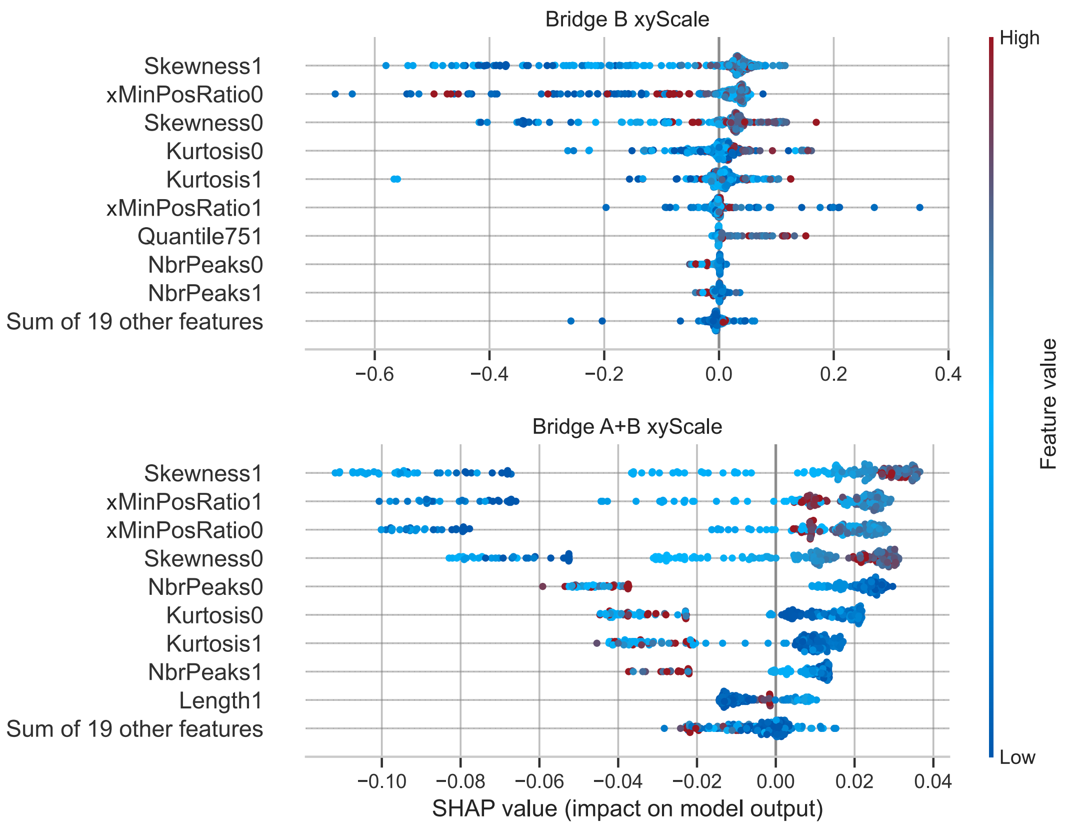


Fig. 5.10.: Beeswarm plot for the RF on Bridge B and `xyScale` (top) and Bridge A+B and `xyScale` (bottom).

event duration is more clear as can be seen in Figure 5.5. Thus, the signal length might be an even more dominant factor. The transfer seems challenging for RF and MiniRocket trained with `xyScale`, since even OA and RC are around 50 %. Only PCA RF has values of over 80 % and even a BA of 74 %. As with five reflectors for Bridge A, the PCA appears to extract robust features. Considering that both bridges have unequal numbers of fields and are of different size, this is a promising result regarding transferability.
Training on both, bridges leads to results comparable to the ones obtained on solely Bridge A. Here, too, only minor differences between `None` and `xyScale` are to observe. This contrasts with the poorer results in the blind `xyScale` transfer to Bridge B.

In order to achieve a more in-depth analysis regarding blind transferability and how the model handles unseen data, we utilize SHAP again (see Figure 5.10). For comparability with Figure 5.8, we analyze the RF although it achieves the worst BA on Bridge B `xyScale`. First, we compare the upper blind transfer plot with Figure 5.8. One aspect to highlight is that mostly the same features are important during in-

ference. At the same time, the RF now values “Quantile75” of the second reflector very highly. A feature which does not have a top nine importance for Bridge A data. Conversely, the length feature does not achieve a place in the top nine for Bridge B `xyScale`. One would have expected otherwise as Bridge B has a more seperate distributions in this regard (see Figure 5.4). Then again, a lower importance of the signal length is what we aimed for with our data augmentation.
Comparing the lower and upper plot, similar features appear in the most important top nine. A striking difference lies within the scale of the SHAP values. The Bridge A RF reaches SHAP values of more than -0.6. The absolute SHAP value of most data points of the Bridge A and Bridge B RF are smaller than 0.1. Seperately they contribute little towards a multi event prediction. Both “Skewness” and “xMinPosRatio” features contribute the most within a similar range. Although it is not possible to draw a definite conclusion, all models with data augmentation exhibit resemblance in the features they value highly. In sum, SHAP indicates that a developed model can extrapolate to new bridges, but as Bridge A and Bridge B show similar behaviour this needs to be studied on other structures.

5.4.3 Conclusion and Outlook

This section introduced a data-driven ML approach to classify multi-variate GBR time series from bridge vehicle crossings. The GBR data originates from two bridges in Germany, which a UAV also monitors as ground truth. One bridge has been used for training, validation, and testing, whereas the second bridge only served as a test set to evaluate transferability. We also investigated the effect of varying measurement points as input. The classification goal is to differentiate between single- and multi-presence events. Particularly challenging is the variable length of the time series and the correlation of the length with a class. For this purpose, we have examined the potential of data augmentation. Methodically, we implement and apply three different ML approaches. We use an RF with 14 hand-crafted features per reflector as input, an RF that uses the output of a PCA as input. Finally, we compare their performance to MiniRocket, a state-of-the-art ML model in time series classification. It shows that MiniRocket outperforms the other models when applying data augmentation to tackle biases. With a BA of $90\,\%$, it achieves satisfying results. We also use SHAP to investigate the effect of data augmentation on the RF. Concerning transferability to an utterly unknown bridge, MiniRocket needs to catch up in BA compared to PCA RF and has only a BA of $74\,\%$. Keeping the second bridge out of the training cycle allowed us to test the transferability and reduced the training size, making it more challenging for the models. SHAP showed that the RF model uses similiar features when being applied to an unknown dataset. When combining both dataset the models performed similar to the ones trained on Bridge A. In sum, we showed that a data-driven classification of GBR data is possible and single events can be identified, as BWIM often focuses on them.

5.5 Vehicle Count Estimation

To achieve a more comprehensive traffic analysis, multi events should not be discarded. In this section, we evaluate approaches to estimate the vehicle count for an event as preliminary work for multi event BWIM. To this end, we investigate both classification and regression. We abstain from training only on Bridge A, as there are only five data points in the `4+ vehicles` class. After splitting the dataset in train and test data, too few samples would be left for five-fold crossvalidation.

5.5.1 Methodology

This section will describe our methodological approach as depicted in Figure 5.11. First, we will describe preprocessing and data augmentation. Afterwards, the feature extraction step is explained. Finally, our ML models concerning our classification and regression tasks are introduced. For comparison purposes, we orientate ourselves at the methodology of Section 5.4.

Preprocessing and Data Augmentation

We reduce the time series data preprocessing to a minimum. The only step we apply is to remove the offset for each sequence. This is necessary as a long-term drift occurs during GBR measurement campaigns [5]. We also do not remove bridge oscillations as they often correlate with the presence of heavy vehicles.

We test all our approaches on two event types. First, we use all available events as long as an event is in a not disturbed mode. In a second step, we build on the results of Section 5.4. Assuming that we can distinguish between single and multi events, it would be enough to consider multi events only for our vehicle count estimation. Disregarding all single events would reduce the dataset imbalance but it also drastically reduces its size.

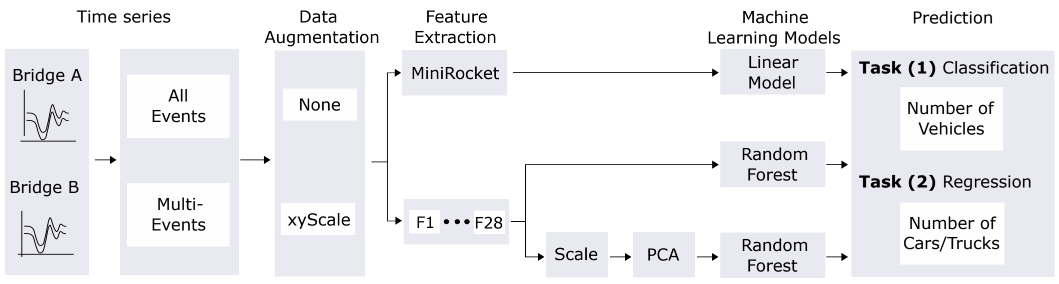


Fig. 5.11.: Schema of our methodological approach for vehicle count estimation. Adopted from [6]

For data augmentation, we use a combination of x- and y-scaling (`xyScale`) and no augmentation at all (`None`). Apart from increasing our dataset, we try to tackle its skew as described in Section 5.3 with data augmentation. x-scaling corresponds to down- and oversampling along the temporal axis. For events with one vehicle, we oversample each event by factors two and three, as single events are often shorter. Conversely, we downsample events with more than one vehicle by the same amount. y-Scaling is done afterward for all sequences by rescaling them to a range of $0.1\,\mathrm{mm}$ to $4.0\,\mathrm{mm}$. From each sequence, five new sequences are generated through y-scaling.

Feature Extraction

Three different approaches for feature extraction are evaluated. First, we use manually developed features listed in Table 5.3. They are extracted for each reflector time series. Thus, with 14 features and two used reflectors, we generate 28 features per event. We also apply scaling and PCA as a second approach to reduce the dimensionality. We transform our 28 features onto eight components since they explain over 95 % of the variance for the Bridge A dataset without data augmentation [9]. Again, we exploit MiniRocket for automatic feature extraction as detailed in Section 3.2.2.

Machine Learning Models

We investigate two different tasks in this study: **Task (1)**, the classification of events according to the number of vehicles within, and **Task (2)**, the prediction of cars and trucks within an event. The first task is treated as a multi-class classification task with the classes `1 vehicle`, `2 vehicles`, `3 vehicles`, and `4+ vehicles`. We implement the second task as a multi-output regression by predicting a value for both car and truck. Since regression can produce floating numbers, but vehicles only occur in natural numbers, we round the prediction to the nearest integer. We use regression since the wide varity in vehicle combinations would lead to many small classes.

The manually crafted features are inputted to an RF. Concerning MiniRocket and **Task 1**, we follow the suggestion of Dempster et al. [35] to use a logistic regression model for `xyScale` augmentation as there are more than $10\,000$ samples in the training set and a ridge model otherwise, meaning for `None` augmentation. However, in the case of the regression task, we will always utilize a ridge regression model.

We apply grid search with 5-fold cross-validation for hyperparameter optimization during training. We split our dataset in a $80:20$ manner for training and testing. Due to class imbalance in **Task (1)**, we apply class weights and use BA as the score for a grid search. The MAE is the scoring metric for the regression task. We do not optimize the hyperparameters for our MiniRocket approach. Instead, we follow the suggestions of Dempster et al. [35] and use default parameters otherwise.

5.5.2 Results and Discussion

This study aims to investigate the potential of ML to count vehicles in GBR bridge crossing events. In this section, we will state and discuss the results of all approaches. The classification task models are evaluated based on BA, OA, P, and RC. Regression performance is expressed by the R^2, MSE, and MAE. The results for both tasks have been combined in Table 5.5.

Regarding **Task 1**, the classification of events according to their number of contained vehicles, RF with no data augmentation outperforms the other models with a BA of $80.4\,\%$. Its confusion matrix can be seen in Figure 5.12. MiniRocket comes second with a BA of $70.2\,\%$. Regarding all events, OA, P, and RC are very high for all models independent of the data augmentation method. This is mainly due to the imbalance of the dataset. Models tend to predict single events because the dataset is heavily skewed towards this class. While both RF and MiniRocket decrease in BA for `xyScale` compared to `None`, PCA RF performs similarly in both cases. This indicates that scaling adds no new information that linear transformation methods can extract, but also, the PCA makes this pipeline more robust. Overall, the PCA RF BA improves slightly when data augmentation is applied. Conversely, RF decreases in BA for `xyScale`.

Tab. 5.5.: Test results for all **Task 1: Classification** and **Task 2: Regression**. The best results concerning BA respectively R^2 for each event type are highlighted. Adopted from [6].

Event Type	Data Augmentation	Model	Task 1 OA in %	Task 1 P in %	Task 1 RC in %	Task 1 BA in %	Task 2 R^2	Task 2 MSE	Task 2 MAE
All events	None	RF	95.1	94.7	95.0	**80.4**	0.74	0.14	0.10
		PCA RF	92.7	92.0	92.7	66.4	0.75	0.14	0.11
		MiniRocket	93.5	93.0	93.4	70.2	**0.80**	0.11	0.08
	xyScale	RF	92.3	92.1	92.3	63.8	0.35	0.25	0.22
		PCA RF	88.5	90.2	88.5	67.9	0.50	0.20	0.18
		MiniRocket	90.8	90.1	90.8	62.3	0.48	0.21	0.19
Multi events	None	RF	81.1	81.1	81.1	67.1	0.64	0.65	0.30
		PCA RF	70.2	70.9	70.2	50.0	**0.66**	0.54	0.30
		MiniRocket	72.9	78.4	72.9	58.3	0.53	0.69	0.36
	xyScale	RF	70.3	70.0	70.3	66.2	0.31	0.68	0.46
		PCA RF	70.2	68.7	70.2	56.9	0.21	0.73	0.54
		MiniRocket	78.4	78.9	78.4	**72.7**	0.25	0.77	0.55

The confusion matrix of RF in Figure 5.12 can be understood as a single vs. multi event classification when combining the classes 2, 3 and 4+ into one class. In this case, we achieve a BA of 90.3 % slightly outperforming MiniRocket as the best model from Section 5.4. Not only that, but the RF also has a higher class resolution. Interestingly, FP and FN for `1 vehicle` are only shared with `2 vehicles`.

Figure 5.13 shows the beeswarm plot for the `None` RF. As this is a multi-class classification task and for each class the SHAP values vary, we limit ourselves to the classes of single vehicle events and events involving two vehicles. It stands out, that unlike the single event classification without data augmentation in Figure 5.8, the RF is not solely depending on the signal length. Both length features are still of highest importance, but other features have consinderable impact too. Even the 19 other features score high SHAP values. It seems that the high scoring of the RF is not only due to the imbalance of the dataset. Hence, instead of applying data augmentation, rephrasing a ML task is another way to force models to learn more complex relations. Again, our "xMinPosRatio" feature is among the top three features regarding importance. Conversely, "NbrPeaks" does only play a small role for those classes. The `2 vehicles` class has a similar importance ranking, but with "Median1" on fourth place. This feature has not appeared in the first 9 in any model of Section 5.4. Quantiles also have more relevance compared to the previous chapter.

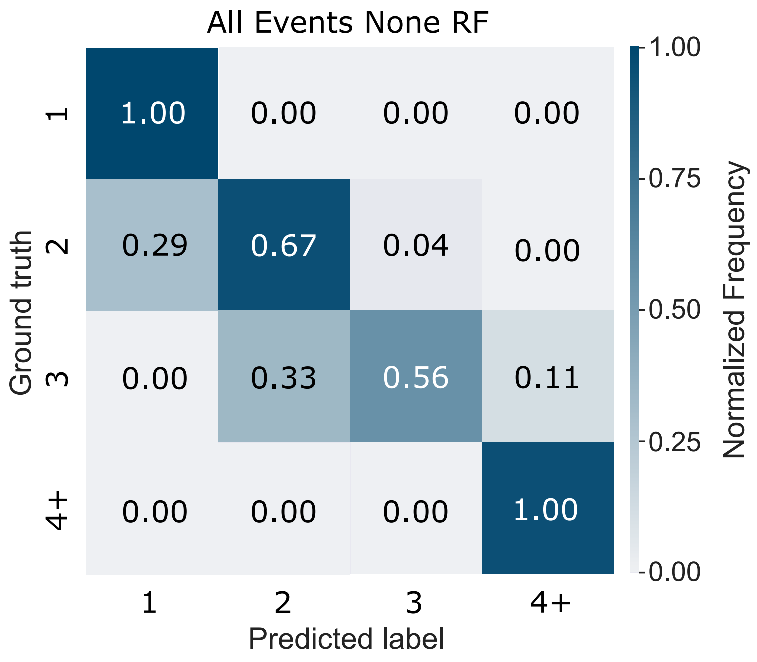


Fig. 5.12.: Confusion matrix for RF with all events and without data augmentation. Adapted from [6] by adding colored scaling along the ground truth axis.

Overall, our feature selection seems to be appropriate but needs fine tuning to reduce dependencies.

Using only multi events in **Task 1** leads to considerably worse results, especially OA, P, and RC. Since no single events are present in the dataset, it is much more balanced than all events. Therefore, the models cannot simply predict one vehicle. MiniRocket outperforms all other models for this event type and `xyScale` with a BA of 72.7 %. It even improves its BA compared to all events. Figure 5.14 shows its confusion matrix. When comparing it to the confusion matrix of RF, it shows that including single events also leads to a better performance for the multi event classes (see Figure 5.12). Thus, estimating vehicle count without data augmentation using an RF outperforms first classifying in single and multi events and then using only multi events for vehicle count estimation.

PCA RF always has the worst BA except for all events `xyScale`. In Section 5.4, we state that when only training on one bridge and predicting on another one, PCA RF achieved the best results for single- vs multi-presence classification. However, with two bridges in the training dataset, PCA RF seems to have issues extracting helpful features. One reason could be that in the Bridge B dataset, events with several vehicles take considerably longer than single events (see Figure 5.5), making it difficult to generalize using PCA. On the other hand, training and testing on

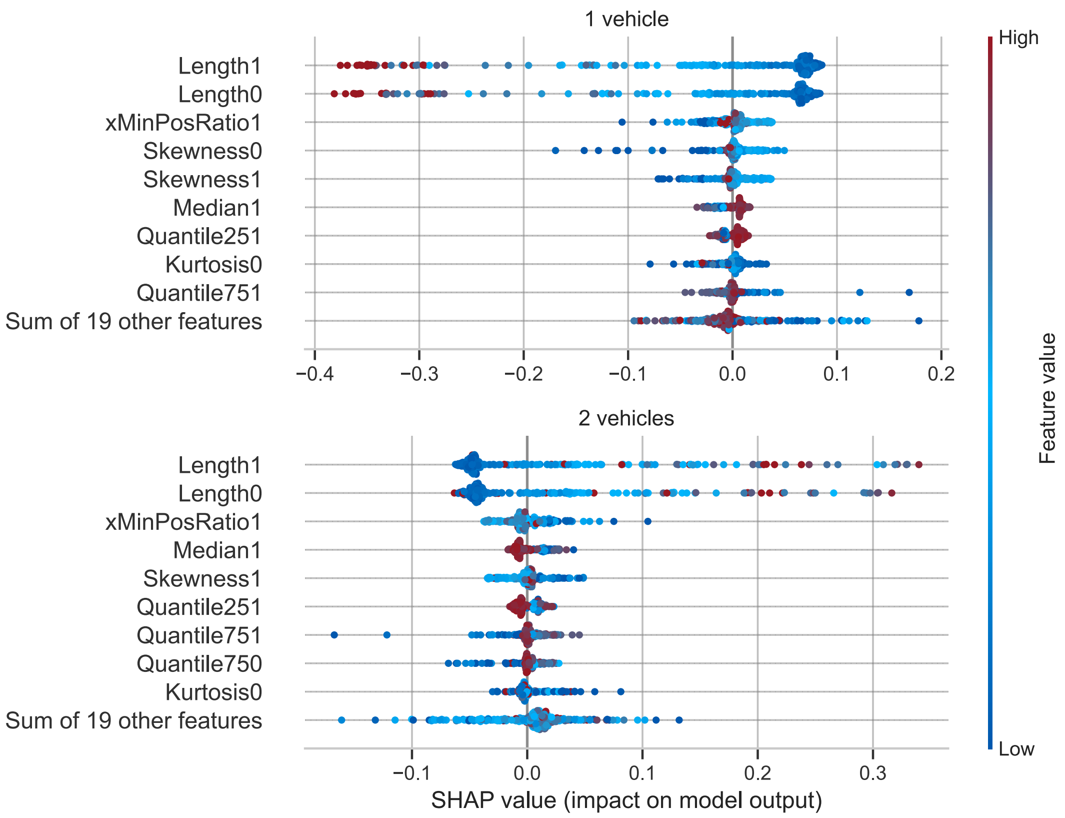


Fig. 5.13.: Beeswarm plot for the RF on `None` data augmentation for the classes `1 vehicle` and `2 vehicles`.

both bridges lead to comparable results with and without data augmentaiton. Here, however, the data augmentation leads to poorer performances.

The results of **Task 2** are also stated in Table 5.5. This task aims to estimate the number of cars and trucks in an event separately using multi-output regression. We have rounded the regression results to the nearest natural number before evaluation. This might skew our results. However, it is closer to reality. MiniRocket achieves the best overall results with a R^2 of 0.8 using all events without data augmentation. In this setup, RF and PCA RF have comparable results. Also, MSE and MAE are close to each other, indicating that the number of prediction outliers is small. PCA RF achieves the best results after removing single events from the dataset with a R^2 of 0.66. RF performs comparably. However, MiniRocket falls considerably behind with a R^2 of 0.53. It is barely better than predicting the mean of the dataset, which is the case for an R^2 of 0.5. The decrease in performance of all models between all events and multi events indicates that the imbalance of the dataset regarding the overwhelming majority of one-vehicle events leads to good results for all events. Also, both MSE and MAE increase due to this aspect. A more detailed analysis using SHAP would be necessary though, as it indicates contrarily on **Task 1**.

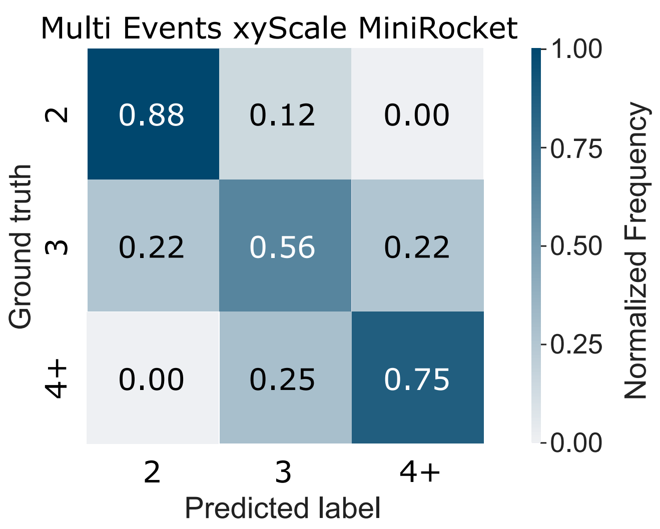


Fig. 5.14.: Confusion matrix for multi events MiniRocket with data augmentation. Adapted from [6] by adding colored scaling along the ground truth axis.

Regarding the effect of data augmentation, it shows that the performance drops heavily when using the `xyScale` dataset. For all events and only multi events, the R^2 for all models is smaller or equal to 0.5. MiniRocket only achieves a R^2 of 0.25 for multi events with data augmentation. Why models decrease in performance when using only multi events needs to be studied more thoroughly as the task appears less complex. One aspect could be the much smaller dataset.

5.5.3 Conclusion

In this section, we discussed ML approaches for data-driven determination of vehicle count from GBR bridge displacement time series data. Two bridges in Germany have been monitored over several days. Together with UAV data for ground truth, a database has been built. With this database, we investigated the potential ML approaches to (1) classify events according to the number of involved vehicles and (2) extract the exact number of cars and trucks in an event via regression. To this end, we used two measurement points per bridge. To simplify the classification task, we grouped all events with four or more vehicles in one class. We investigated the effect of applying data augmentation and using only multi events in the dataset.

Methodologically, we implemented three different ML approaches, which can handle variable-length time series data. First, we manually crafted features extracted from the time series data and passed it to an RF. Second, we scaled those features and reduced their dimensionality via PCA before passing the resulting eight components to an RF. Finally, we exploited MiniRocket, which extracts features by applying convolutional kernels to the time series data. It showed that RF achieves the best results for the classification with a BA of $80.4\,\%$. With SHAP we could show that the RF relies on more features than only the signal length.

When removing single events and applying data augmentation, MiniRocket outperformed all other models. It could classify multiple events according to their vehicle count with a BA of $72.2\,\%$. Still, the RF with no augmentation outperformed the multi event only models.

Similar observations could be drawn from our regression approach, as no data augmentation gives the best results. Using all events and applying no data augmentation, MiniRocket achieved R^2 of 0.80. PCA RF had the best R^2 of 0.66 with only multi events. Data augmentation worsened the results for all events and only multi events.

In sum, we showed that a data-driven classification and regression approach for vehicle count determination is feasible. These promising results can lead to more sophisticated methods for GBR-based BWIM than only focusing on single events as it is currently done, as more detailed information about vehicle counts is acquired.

5.6 Conclusion

In this chapter, we developed a new data-driven method to estimate vehicle counts in GBR bridge crossing events. One major challenge is the varying length of the time series dataset as well as the correlation between signal length and class affiliation. Thus, we investigated models which can handle such data. For one, MiniRocket extracts features itself by applying random convolutions. Furthermore, we performed feature engineering and tested two RF, once in combination with PCA. To tackle our small and imbalanced dataset, we also examined data augmentation.

In a first evaluation, we simply tried to distinguish between single and multi events. Multi events include simultaneous crossings of at least two vehicles. We trained and tested on Bridge A and only tested on Bridge B to investigate blind transferability. We showed, for one thing, that the data augmentation can help achieving better results, as the models otherwise primarily depend on the signal length for their classification. To this end, we exploited SHAP, an ad-hoc model-agnostic method for interpretability in ML. Overall, the MiniRocket outperformed all other models using data augmentation and two reflectors achieving a BA of $90\,\%$. Including all five reflectors of Bridge A does not improve the results and jeopardizes transferability. We tested all models on the unknown Bridge B showing that, while transferability is possible, it is also not reliable. Combining both datasets lead to results similar to the only Bridge A usecase.

The second evaluation raised the challenge by training the models to estimate the exact number of vehicles involved in an event. This time, the datasets for both bridges were used for training. RF trained without data augmentation outperforms all other models with a BA of $80.4\,\%$. Using SHAP we could show that the model does not solely rely on the signal length against our expectations. Our data augmentation proved to be unnecessary in this regard. Additionally, we evaluated models only on multi events, presuming a succesfull single event classification as per the previous section. Overall however, the best RF trained on all events came out victorious again. Finally, we try to obtain the exact number of cars and trucks within an event using regression. Here, MiniRocket prevailed with an R^2 of 0.8.

In sum, this chapter investigated the challenges of variable-length multi-variate time series classification, showing that in the context of GBR displacement, data-driven solutions can be found. This finding is a first step towards our **Research Goal II** of classifying bridge crossing events and the vehicles causing these events. As mentioned before, many BWIM approaches are based on single vehicle crossings. By performing vehicle count estimation as done in this chapter, following studies can now rightly focus on single events for extracting vehicle information from bridge crossings.

6 Vehicle Classification

Taxonomy is described sometimes as a science and sometimes as an art, but really it's a battleground.

— **Bill Bryson**
(Author and Journalist)

This chapter includes material from the following works:

Matthias Arnold and Sina Keller. "Machine Learning and Signal Processing for Bridge Traffic Classification with Radar Displacement Time-Series Data". en. In: *Infrastructures* 9.3 (Mar. 2024). Number: 3 Publisher: Multidisciplinary Digital Publishing Institute, p. 37

It is cited as [8] and marked with a green line.

6.1 Introduction

The ultimate objective of BWIM is to estimate the weight of vehicles to either determine the load on the bridge for SHM or to detect overweight vehicles for traffic monitoring. This study will not investigate load estimation since we do not have reference data. However, Ojio et al. [115] showed that it is possible to extract the vehicle load from bridge displacement data. To this end, they use a method developed by Moses [109]. This method requires certain vehicle parameters like the speed to be known beforehand [81, 59]. We hence investigate to what extent such vehicle parameters can be extracted from bridge displacement data. In other words, the central objective of this section is the potential of GBR displacement signals for a remote and data-driven BWIM. Building on the results of the previous chapter, the evaluation concentrates on single events of Bridge A.

Overall, we focus on the two objectives displayed in Figure 6.1:

- We analyze and implement ML approaches to determine relevant BWIM vehicle configurations, including vehicle type, presence of a trailer, lane, locus, speed, axle count, and spacing.
- We investigate signal processing techniques in the context of axle-related parameters.

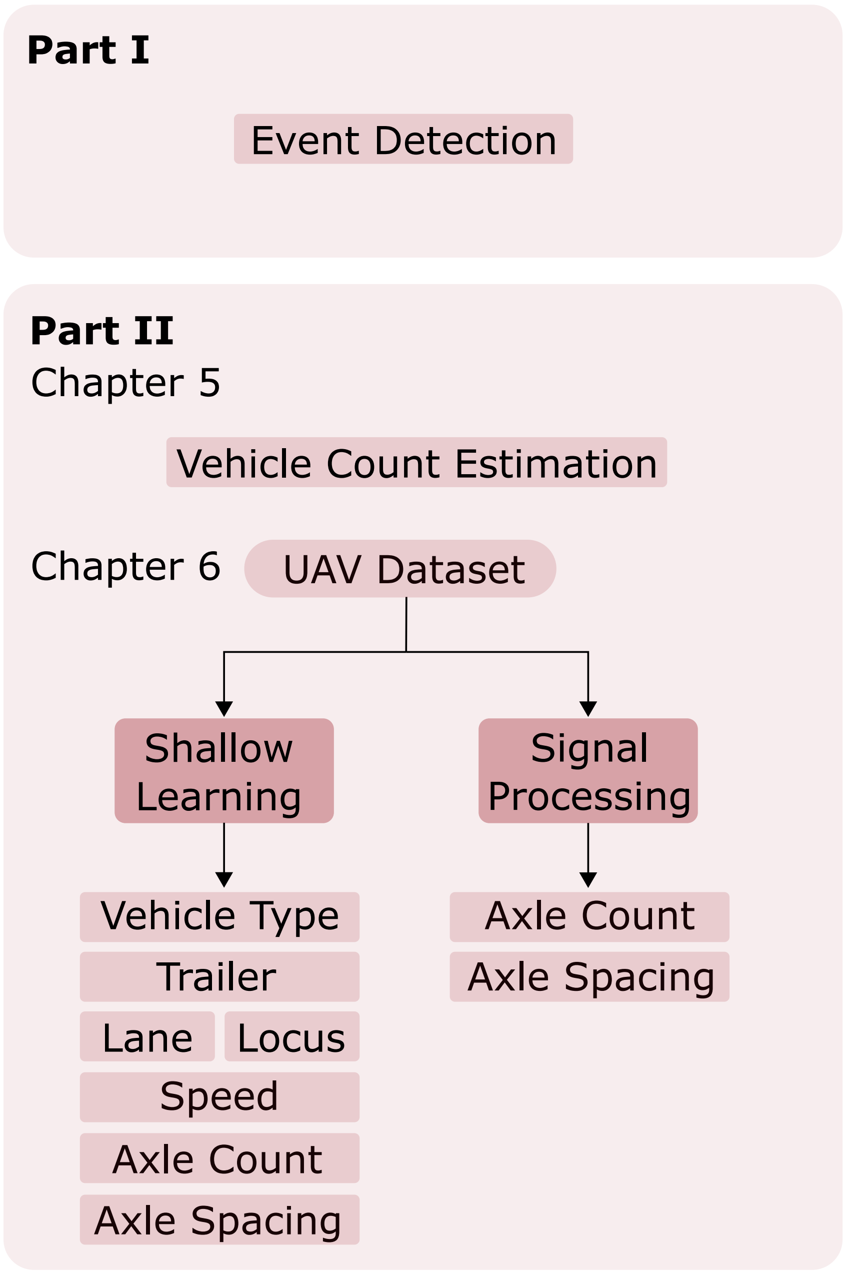


Fig. 6.1.: Overview of the event classification chapter.

We structure our chapter as follows: First, we will describe state-of-the-art studies researching vehicle parameter extraction in the context of BWIM in Section 6.2. The Bridge A UAV dataset used in this chapter is detailed in Section 6.3. Afterwards, we describe the methodology for each parameter (see Section 6.4). The results for all explorations are stated in Section 6.5 and discussed in Section 6.6. Finally, Section 6.7 summarizes this chapter's findings and draws a conclusion.

6.2 Related Work

As we detail the state-of-the-art time series classification approaches concerning variabe-length sequences in Section 5.2, we refer the reader to that section for more information. In this section, we will focus on the extraction of vehicle parameters in the context of BWIM. An overview of BWIM studies relevant to this work is presented in Table 6.1.

Several studies used ML or DL methods to extract vehicle information from NOR BWIM time series data. Kawakatsu et al. [73] attached a single strain sensor to the span of a $300\,\mathrm{m}$ long-concrete bridge in Japan. They used CNNs to detect vehicles and estimate the speed, locus, and axle count. As input, $8\,\mathrm{s}$ windows of strain data are passed to the network. They use a traffic surveillance system for their [... reference] data, resulting in up to $996,093$ samples depending on the classification task. In [74], they expand their dataset by a steel bridge and investigate acceleration sensors as an alternative to strain sensors. The MAE for locus estimation is $0.097\,\mathrm{m}$ or $0.127\,\mathrm{m}$ for strain data depending on the driving direction. Kawakatsu et al. [75] extend their previous work by load estimation using multi-task CNNs. A load meter provides the necessary [... reference] data. They also increase the sequence length of the input data to $20\,\mathrm{s}$. For a $74\,\mathrm{m}$-long steel bridge, they achieve an MAE of $0.92\,\mathrm{m\,s^{-1}}$ for speed, and $0.354\,\mathrm{m}$ for axle spacing. The effects of more than one sensor are investigated in [76]. Eleven sensors are used as input to different CNN architectures for several bridges, further improving the previous results. For a two-span bridge, they achieve a balanced accuracy score of $91.92\,\%$ for an imbalanced lane estimation. Other features are comparable or slightly improved compared to their previous studies.

For axle detection, simple peak detection algorithms are often used [74]. Yu et al. [157] applied a combination of wavelet transform and peak detection on strain data from Finite element method (FEM) simulations. Although they only use FEM data and do not transfer to real-world data, they make several relevant findings. Firstly, it is shown that axle information can be extracted from global bridge responses. Furthermore, the sampling frequency significantly impacts the identification accuracy since it leads to sharper peaks. For example, at $200\,\mathrm{Hz}$, the axle spacing identification errors for a three-axle vehicle traveling at $30\,\mathrm{m\,s^{-1}}$ are $25.3\,\%$ and $97.1\,\%$, respectively. At a sampling frequency of $500\,\mathrm{Hz}$, these errors decrease to $0.2\,\%$ and $1.43\,\%$. Finally, they show that road surface conditions can severely impact the results. Lechner et al. [82] also use wavelets for BWIM based on crack displacement sensor data. They measure the width changes of an existing crack during traffic loading. With this local response, they can successfully obtain vehicle speed, axle count, and distances. Using the influence line, they can also compute individual axle loads. Zhao et al. [161] use free-of-axle-detector (FAD) in combination with wavelets for improved strain-based axle detection. FAD BWIM uses additional FAD sensors attached to the lower side of the bridge. They have shown that axle-induced peaks in the FAD strain signal with Daubechies wavelets are more easily distinguished. While such FAD BWIMs give good results, they need many sensors, leading to a higher probability of failure.

Concerning contactless BWIM, Ojio et al. [115] investigate the potential of cameras for bridge displacement measurements. They can extract the axle loads of a few reference vehicles with known weights from the bridge displacement using the influence line. However, regarding other aspects such as speed, lane, and axle spacing, they rely on a second camera instead of directly extracting these parameters from the displacement data. To the best of our knowledge, no study exists that investigates GBRs for BWIM.

Tab. 6.1.: Studies related to the comparison of extracted vehicle configurations. Adopted from [8].

Parameter	Response, Sensor, or Model	References
Speed	Camera	[115]
	Displacement	[82]
	Strain	[73, 75, 76]
	FEM	[157]
Lane or locus	Strain	[73, 76]
Axle count	Camera	[115]
	Displacement	[82]
	Strain	[73, 75, 161, 76]
	FEM	[157, 161]
Axle spacing	Strain	[75, 76]
	FEM	[157]

6.3 Dataset

During our measurement campaigns, we deployed a UAV to monitor the bridge deck with a sampling frequency of around 25 fps. Furthermore, several control points along the bridge were measured using a [... tacheometer]. This enables us to transform image coordinates in pixels to positions in meters in our radar reference system. Those transformations have been conducted following Juretzko [68] by the Institut für Photogrammetrie und Fernerkundung (KIT-IPF), our partner during ZEBBRA. In Figure 6.2, two control points are highlighted with yellow circles. As they are visible in the UAV video data for each frame, deviations in the UAV position do not influence the transformation. Five measurements have been conducted at Bridge A covering various weather conditions over a span of almost two years. Section 2.2 gives an overview of those campaigns.

To acquire information about locus, axle spacing, and speed, we labeled each axle on three images per vehicle using a labeling tool [148]. Figure 6.2 shows a two-axle bus's labeling results ([... reference] data). For the position of an axle, we used the x-wise center of a bounding box, which is then transformed [... to meters]. The distance between consecutive axles has been calculated via the position difference. By labeling more than one image per vehicle, we determined the speed. For that, we used the positions of the first axle, the number of frames elapsed between each labeled image, and the UAV sampling frequency. Finally, the y position of the lower bound of the first axle represents the locus. Labeling three images per vehicle has allowed us to detect outliers in the extracted properties. Imprecisions during the labeling process can lead to minor variations in the [... reference] data. Furthermore, the transformation accuracy precisely depends on how the control points are recognized during the transformation process. To validate the [... reference] data, we compared datasheets for recognizable vehicles to the extracted axle spacing to verify our procedure. This provided a satisfactory match. Therefore, the margin of these errors is not significant enough to have a relevant impact. One challenge during the labeling process was posed by the darkness in combination with wet streets since it rendered the asphalt and the wheels barely distinguishable. Especially for trucks with raisable tires, it has been challenging to say whether they have touched the street or not.

Table 6.2 lists different vehicle information distinguished by vehicle type. Axle counts exclude vehicles with trailers as we later estimate axle related parameters only on the trailer-free subset. Among others, we drop the one 7 axle truck, simplifying axle-related prediction tasks. 10 cars and 33 trucks have trailers. As expected, cars [without trailers] always have two axles, whereas trucks have varying axle

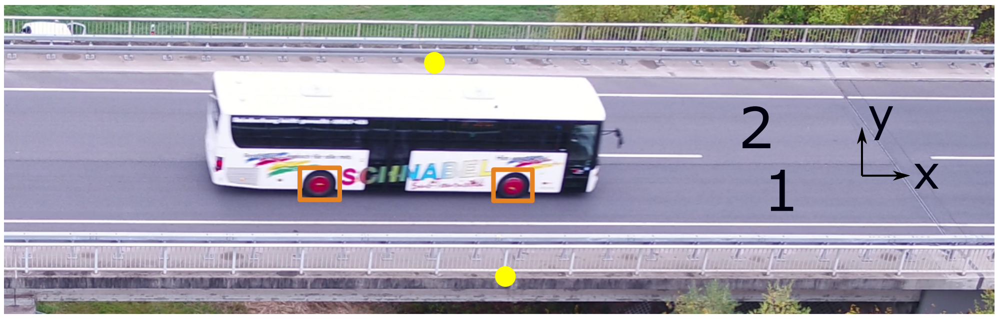


Fig. 6.2.: Example of a labeled vehicle using the tool provided by Tzuta [148]. Bounding boxes have been drawn around the axles, as indicated by the orange bounding boxes. The yellow circles represent two control points for the coordinate transformation. Furthermore, the lane number is shown. Adapted from [8].

Tab. 6.2.: Trailer and axle configuration as well as lane information for single cars and trucks in our Bridge A dataset. Vehicles with trailers are not included in the axle count. Adapted from [8].

Type	Trailer	**2 Axles**	**3 Axles**	**4 Axles**	**5 Axles**	**Lane 1**	**Lane 2**
Car	10	445	-	-	-	235	220
Truck	33	24	8	25	41	78	53

configurations. Overall, 445 single cars and [... 98] single trucks [without trailers] have been registered and labeled. Additionally, Table 6.2 shows the distribution of events for both lanes. Lane 1 has been used more often, but all values are similar.

Figure 6.3 shows the distributions for our regression tasks: speed, locus, and axle spacing estimation. The axle spacing contains all spacings of a vehicle independent of the axle position. Both axle spacing distributions only include vehicles without trailers.
The speed of cars is generally higher and sometimes exceeds the speed limit of $100\,\mathrm{km\,h^{-1}}$. Trucks drive slower on average. The locus shows similar distributions for both vehicle types and corresponds to the existence of two lanes. The most significant difference is shown in the axle spacing, as cars have values of approximately $2.8\,\mathrm{m}$, while trucks have values ranging from approximately [... $1.25\,\mathrm{m}$ to $8.17\,\mathrm{m}$]. At the peak of the car distribution, the truck distribution has a local minimum. If trailers were included, the truck distribution would span from $0.7\,\mathrm{m}$ to $10.3\,\mathrm{m}$ due to some extrem cases.

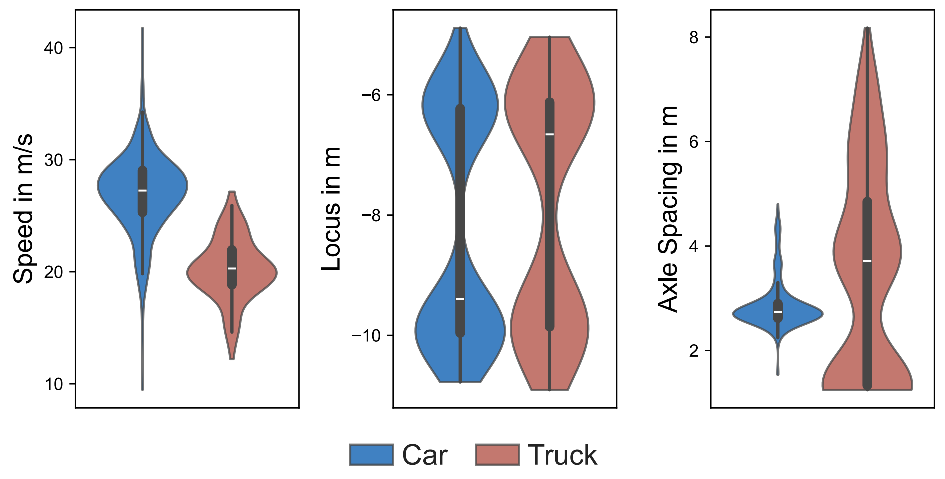


Fig. 6.3.: Distributions of speed, locus, and axle spacing in the Bridge A dataset. by exluding vehicles with trailers from axle spacing and by changing the color palette.

6.4 Methods

In this chapter, we explain our preprocessing and feature extraction steps and, afterwards, the data-driven approaches and methods for each vehicle parameter. First, we investigate the distinction between cars and trucks. For all other tasks, we investigate both vehicle types simultaneously to have a more extensive dataset, as well as only trucks, by disregarding cars, as trucks are more relevant to SHM.

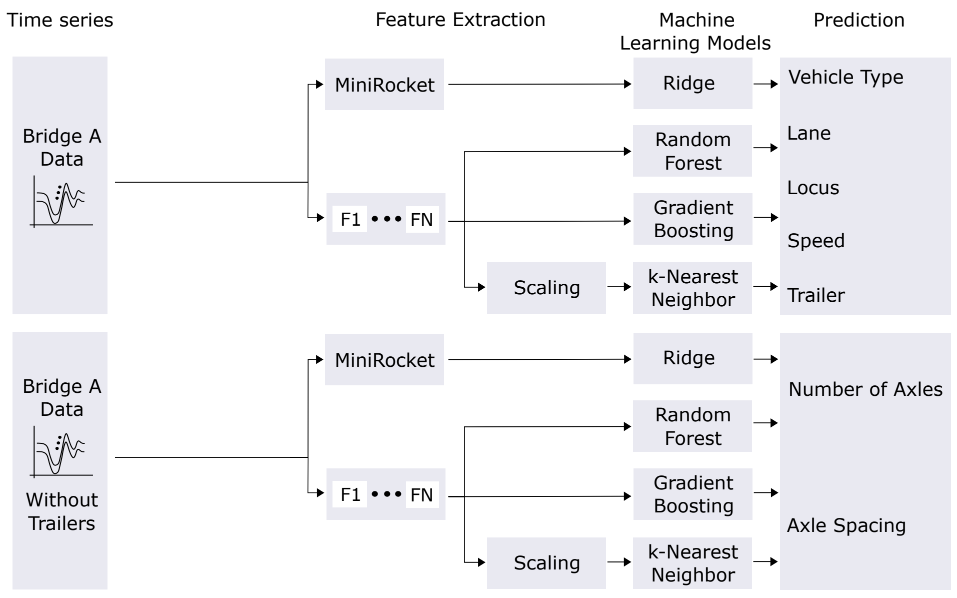


Fig. 6.4.: Methodology for our feature-based methods on Bridge A data. $F1$–FN represent manually extracted features. We use a high-pass filter before feature extraction for the axle counting and axle spacing estimation. Adapted from [8].

6.4.1 Preprocessing and Feature Extraction

We minimize the preprocessing, so removing each time series' offset is the only step except for axle counting and axle spacing estimation, where we use an additional high-pass filter before feature extraction. This offset comes from long-term drifts in the signal due to environmental influences [5]. Otherwise, no filtering is applied since we want to maintain high-frequency information.

Figure 6.4 shows the methodology for our feature-based approaches. Among others, we test various models in combination with manually crafted features. Table 6.3 summarizes all features used in this [... section]. Each feature is calculated for each used time series. Only a part of these features is used for a specific task to avoid making ML predictions more challenging by adding irrelevant features. This selection will be stated in the corresponding subsections. Different input features are passed to the ML models since the number of reflectors also varies depending on the task. Since tree-based models are scale-independent, no scaling is applied to the input features for RF [18] and GB [47]. The input features are scaled in the case of KNN [28].
Again, we investigate MiniRocket as an alternative to manual feature extraction as described in Section 3.2.2.

Tab. 6.3.: These 11 features have been extracted from the GBR time series data. Adapted from [8] by removing the calculation method. Information about how these features are calculated can be found in Section 3.1.

Feature No.	Name of Feature
1	Maximum
2	Minimum
3	Mean
4	Skewness
5	Kurtosis
6	Median
7	Length
8	xMinPosRatio
9	Power
10	MAD
11	GradMax

We split our data in an $80 : 20$ manner for training and testing, and we use stratified sampling for classification tasks to maintain class frequency. During training, we apply a 5-fold cross-validation grid search to find the best hyperparameters for each regression and classification task. For MiniRocket, we use the configuration recommended by the authors, which also includes a ridge regressor or classifier as the final step. During grid search, we optimize the MSE for regression and the BA for classification.

6.4.2 Vehicle Type

Since we distinguish between cars and trucks in the following tasks, we want to investigate whether data-driven distinction is possible for completeness. Although it seems trivial to use a threshold, the driving lane also comes into play. Therefore, we use the reflectors 2 and 4 (see Figure 2.4). As input features, we use 2, 3, and 9 of Table 6.3. So, for each time series, we extract the minimum, mean, and power using the corresponding calculation methods. As we use two reflectors and extract each feature for each reflector, six features are extracted in this task. These features are then directly passed to RF, GB, and KNN after scaling to predict the vehicle type.

6.4.3 Lane and Locus

The offset of the reflectors in the y direction, as depicted in Figure 2.3, makes it possible to discern the vertical driving position of a vehicle. In the first step, the lane shall be determined in a classification task using reflectors 2 and 4. As a baseline, which we will refer to as POWER, we compare the signal power (Feature 9 in Table 6.3) for both reflectors. The lane is determined depending on which one has a higher value. The signal power of reflector 2 is greater, and the vehicle drives on lane 2. The input for the features-based models consists of the features 2, 3, and 9. In addition to the lane, the locus is estimated by regression to have a more precise vehicle localization. We use all 5 reflectors with the same features for locus regression.

6.4.4 Speed

Global responses, such as displacement, require a different approach than usual to calculate the speed since extracting it via peak detection from local responses is impossible. Axles normally cause distinct peaks in the local signals, like strain data. By measuring at two positions the speed can be determined via their distance and the axle travel time inbetween. Such local properties have so far not been discovered in global displacement data. Therefore, we use data-driven ML to extract the vehicle speed. With the results of Part I, we can extract vehicle crossings, but just using the length of such an event is not enough since the vehicle length also plays an important role. To show this, we additionally train a LR for speed estimation only on the signal length as a baseline. The feature-based models are trained with the features 1 to 11 from Table 6.3. As the input time series, we use reflector 3.

6.4.5 Trailer

As mentioned above, investigations regarding axle-related parameters will only take vehicles without trailers into account in order to reduce complexity. In a first step, it is necessary to develop data-driven methods, which can detect the presence of a trailer. To this end, we use reflector 3 and all features of Table 6.3.

6.4.6 Axle Count

We treat the determination of the axle count as a classification problem with either 2, 3, 4, or 5 axles. However, we disregard all events which include trailers as they have a very wide variety of configurations but our dataset is small. With more data, a more precise classification might be possible. All predictions are made using only the signal of reflector 3. Again, we exploit all 11 features.

Apart from using ML, we also investigate the deterministic methods using a bandpass filter (BANDPASS) and CWT (WAVELET) as novel approaches on only the Bridge A dataset. Their pipelines are depicted in Figure 6.5 and Figure 6.6, respectively. They are similar except for the first step of the pipeline, where its corresponding filtering or transformation is applied. As input, they receive unfiltered GBR bridge displacement data from one reflector. The output consists of a list of the positions of the detected peak. For BANDPASS, we apply a forward-backward Butterworth bandpass in the first step. The order of the filter is 50 and the critical frequencies are (45 Hz, 65 Hz). We choose `gaus7` as implemented by Lee et al. [83] and the 3. coefficient for our WAVELET approach. This corresponds to $m = 7$ regarding Equation (3.23). These parameters have been determined as part of the parameter tuning process but outside the grid search. The bandpass-filtered signal and the wavelet transform result are then squared and smoothed by a weighted moving-average filter to obtain the distinguishable peaks. Ultimately, the signals are normalized to their highest value before searching all peaks to achieve generalization over all vehicles. We use the `find_peaks`-method of the Python-package scipy for peak detection [152]. The window size of the moving-average filter and the distance and prominence of a peak during peak detection are regarded as hyperparameters and deduced using the training dataset. Depending on the driving side and thus the driving direction, we discard peaks in half of the signal during which the vehicle does not enter or leave the bridge. [... As we will show in Section 6.5], only the entering or leaving process is relevant for peak detection. Finally, we treat the length of the list of detected peak positions as the axle count. If no peak or only one is detected, we assume two axles.

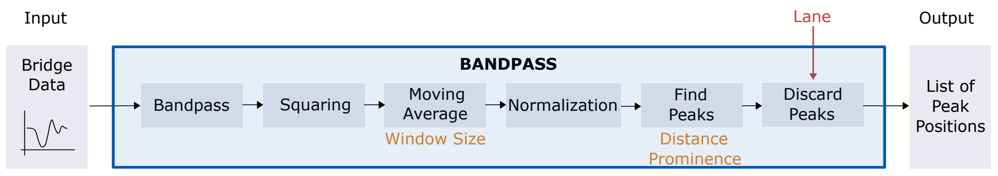


Fig. 6.5.: Pipeline for our BANDPASS approach for axle configuration extraction. The orange-colored hyperparameters are determined during the grid search. The lane information is used in the `Discard Peaks` step as an additional input parameter to filter the detected peaks. Adapted from [8] by changing the color palette.

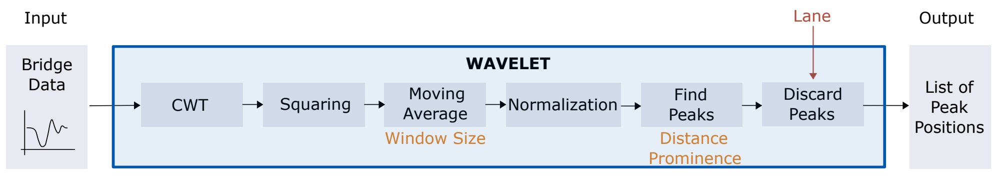


Fig. 6.6.: Pipeline for our WAVELET approach for axle configuration extraction. The orange-colored hyperparameters are determined during the grid search. The lane information is used in the `Discard Peaks` step as an additional input parameter to filter the detected peaks. Adapted from [8] by changing the color palette.

6.4.7 Axle Spacing

Finally, we investigate how well the distance of axles can be determined. We treat this as a multi-output regression. Our BANDPASS and WAVELET procedures return a list of detected peak positions [... $\underline{x}$] (see Figures 6.5 and 6.6). A peak at position i in the list is interpreted as axle number i. The temporal distance between the peaks i and j can be calculated by subtracting the consecutive positions x_j and x_i and then divided by the GBR sampling rate of 200 Hz. With the speed v_{UAV} from the UAV data, this can finally be transformed into axle spacing d_{ij} between axle i and j according to

$$d_{ij} = \frac{x_j - x_i}{200\text{ Hz}} \cdot v_{UAV}. \tag{6.1}$$

We assume the speed can be correctly extracted from Section 6.4.4. We also assume that the axle count is known and ignore additionally detected spacings during evaluation. The same goes for our ML approaches. Like for axle count estimation, we rely on reflector 3 and use all 11 features of Table 6.3.

6.5 Results

In this section, we will state the results of our methods described in Section 6.4 in a corresponding order. Each approach except vehicle type classification is evaluated once for all vehicles and once for only trucks since they are more relevant for SHM. Classification tasks are evaluated based on BA, OA, P, and RC. Regression performance is expressed by the R^2, RMSE, and MAE. The results of our filtering approaches will be addressed in detail in Section 6.5.5.

6.5.1 Vehicle Type

The results for the distinction between cars and trucks, which is relevant for SHM, are displayed in Table 6.4. All feature-based models classify the vehicle type perfectly. Using the raw time series data, only MiniRocket misclassifies one truck as a car.

Tab. 6.4.: Classification results for vehicle type estimation on Bridge A data with reflector 2 and 4. The highlighted values represent the best results for each vehicle group. Adopted from [8]

Model	BA **in %**	OA **in %**	P **in %**	RC **in %**
MiniRocket	98.1	99.2	100	96.1
RF	**100**	100	100	100
GB	**100**	100	100	100
KNN	**100**	100	100	100

6.5.2 Lane and Locus

We investigate both lane classification (see Table 6.5) as well as locus regression (see Table 6.6), which is the lateral position of a vehicle. MiniRocket provides the best results for both vehicle groups with 94.3 % and 97.9 %. Except for RF, all models have a BA of over 90 %. POWER shares the first place with MiniRocket in the case of only trucks. Both approaches misclassify only one truck, leading to an RC of 100 %. Yet, RF, whose concept relies on comparing input values, has the worst

Tab. 6.5.: Classification results for lane estimation with reflectors 2 and 4 of Bridge A. The highlighted values represent the best results for each vehicle group. Adopted from [8]

Model	All Vehicles				Trucks Only			
	BA in %	OA in %	P in %	RC in %	BA in %	OA in %	P in %	RC in %
POWER	92.7	92.4	87.1	98.2	**96.9**	96.3	91.7	100
MiniRocket	**94.3**	90.0	98.2	94.1	**96.9**	96.3	91.7	100
RF	83.2	83.1	79.7	85.4	89.2	88.9	83.3	90.1
GB	93.3	93.2	91.2	94.5	93.8	92.6	83.3	90.1
KNN	93.4	93.2	89.8	96.4	92.3	92.6	90.1	90.1

Tab. 6.6.: Regression results for locus estimation with all five Bridge A reflectors. The highlighted figures represent the best results for each vehicle group. Adapted from [8] by replacing MSE with RMSE.

Model	All Vehicles			Trucks Only		
	R^2	RMSE in m	MAE in m	R^2	RMSE in m	MAE in m
LR	0.57	1.22	1.06	0.91	0.61	0.47
MiniRocket	0.84	0.75	0.54	0.96	0.42	0.36
RF	0.59	1.18	0.79	0.32	1.69	0.99
GB	0.80	0.83	0.57	0.91	0.63	0.49
KNN	0.81	0.81	**0.50**	0.99	0.04	**0.16**

results for both cases. The difference between both vehicle selections is slight, but the performance for only trucks is better overall.

A more fine granular lateral position estimation via the locus using all five reflectors resulted in Table 6.6. Again, MiniRocket beats all other models for all vehicles, with an R^2 of 0.84. When discarding cars, KNN has an almost perfect R^2 score of 0.99, slightly surpassing the MiniRocket with 0.96. All models improve from all vehicles to only trucks, like in lane classification, except for RF, which achieves worse results.

As we can use SHAP on GB, we do so to investigate the model behaviour. Figure 6.7 shows the beeswarm plot as introduced in Section 5.4.2. The features' names are zero-indexed. In other words “Power3” refers to the power feature of reflector 4. The expected locus value is $8\,\mathrm{m}$ which fits its distributions of Figure 6.3. According to the feature importance, the model ranks features of reflector 1 and 4 very highly. Especially the signal power of reflector 4 can shift the prediction by $1.5\,\mathrm{m}$ in either direction. The wide variance in the sum of remaining features indicates that other reflectors play an important role, too.

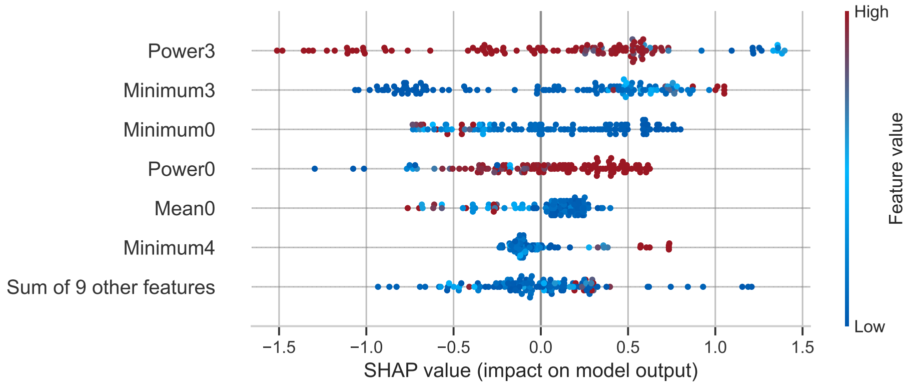


Fig. 6.7.: Beeswarm plot for locus estimation using GB.

Tab. 6.7.: Regression results for speed estimation with reflector 3 of Bridge A. The highlighted values represent the best MAE for each vehicle group. Adapted from [8] by replacing MSE with RMSE.

Model	All Vehicles			Trucks Only		
	R^2	RMSE in m/s	MAE in m/s	R^2	RMSE in m/s	MAE in m/s
LR	0.71	2.25	1.81	0.67	1.49	1.22
MiniRocket	0.94	1.03	**0.76**	0.71	1.39	**1.02**
RF	0.85	1.64	1.12	0.53	1.78	1.25
GB	0.85	1.60	1.14	0.75	1.29	1.06
KNN	0.85	1.61	1.14	0.70	1.42	1.13

6.5.3 Speed

Table 6.7 shows the results for the speed regression task. MiniRocket achieves the best performance for both vehicle sets. For all vehicles, it achieves an R^2 of 0.94 and an MAE of $0.76\,\mathrm{m\,s^{-1}}$, which is slightly better than the $0.86\,\mathrm{m\,s^{-1}}$ of Kawakatsu et al. [76]. All feature-based methods have comparable results for R^2 and are less effective than MiniRocket. LR, on only the duration of the bridge crossing event, has the worst results but still is not far off from MAE. Its RMSE, however, is considerably worse compared to the other models.

LR does improve regarding [... RMSE] and MAE when only using trucks, although R^2 decreases. The same behavior can be observed for KNN and GB. GB has the best R^2 and RMSE for trucks and is slightly worse than MiniRocket in MAE. MiniRocket, with an MAE of $1.02\,\mathrm{m\,s^{-1}}$ is still only $0.16\,\mathrm{m\,s^{-1}}$ worse than Kawakatsu et al. [76]. However, they evaluate more data, making their results more robust.

6.5.4 Trailer

The results for detecting trailers, or rather events without trailers for axle-related tasks, are listed in Table 6.8. Independent of the vehicle group, MiniRocket outperforms all other models. Furthermore, its BA stays consistant with $82.4\,\%$ when including cars and $83.2\,\%$ otherwise. Both of its confusion matrices are given in Figure 6.8.
RF attains similar but slightly worse values. GBs BA drops heavily when removing cars from the dataset. Finally, KNN attributes almost all trailer-including events to the wrong class. Accordingly, the BA is less than $60\,\%$.

Tab. 6.8.: Classification results for trailer detection with reflector 3 of Bridge A. The highlighted values represent the best results for each vehicle group.

Model	**All Vehicles**				**Trucks Only**			
	BA **in %**	OA **in %**	P **in %**	RC **in %**	BA **in %**	OA **in %**	P **in %**	RC **in %**
MiniRocket	**82.4**	95.8	97.3	98.2	**83.2**	88.9	90.5	95.0
RF	81.1	74.6	98.8	73.4	80.0	76.2	66.7	54.3
GB	70.8	93.2	95.5	97.2	49.6	66.7	73.9	85.0
KNN	54.6	91.5	93.0	98.2	59.3	74.1	78.3	90.0

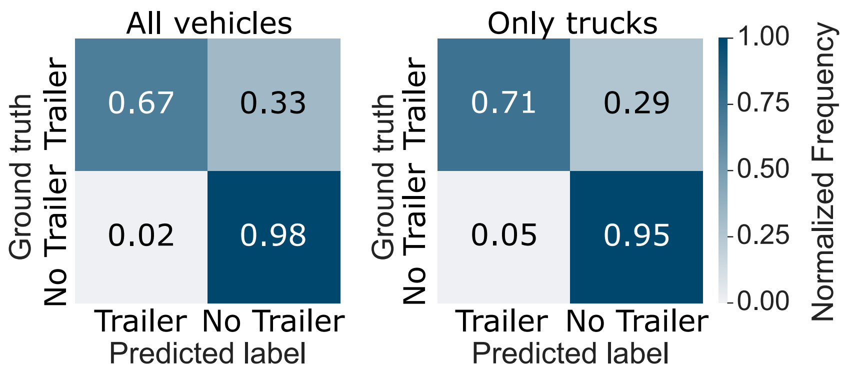


Fig. 6.8.: Confusion matrices for MiniRocket on the trailer detection task for both vehicle groups.

6.5.5 Axle Count

As explained in Section 6.4.6, we investigate the potential of bandpass filtering and wavelet transform to detect axles in GBR bridge displacement data. Three examples of these approaches can be seen in Figures 6.9 to 6.11. More examples are attached in Appendix A.3. In all images, first, the vehicle is depicted, and then the results of the filtering and the smoothed time series. The filtered signal corresponds to the intermediate result after the bandpass filtering and CWT in Figures 6.5 and 6.6, respectively. The envelope represents the unitless waveform after the normalization step from which the peak positions are acquired. We have marked the peaks, which have been detected with the respective approach and the learned hyperparameters. Finally, starting from the first detected peak, positions where we would expect subsequent peaks due to the labeled UAV data are shown.

Figure 6.9 shows the bus with two axles from Figure 6.2. The bandpass filtered signal has two areas with a high amplitude vibration in the frequency range of 45 Hz to 65 Hz. Accordingly, the envelopes for both approaches have two peaks. Assuming that the first peak corresponds to the first axle, the second peak matches the expected position. A more complex example is depicted in Figure 6.10 since the truck has five axles, three of which are very close to each other. Again, these axles are recognizable and at their expected position in the bandpass signal and its envelope. However, one additional peak does not belong to any axle. This peak does not exist in the wavelet signal. Conversely, the first peak is missed, and the final three axles are combined into one axle group in the signal. Finally, Figure 6.11 shows a truck with a similar axle configuration. Seven peaks are detected using the BANDPASS approach. The final three axles are again registered as a single axle group. WAVELET clearly shows the first two axles at the expected positions, yet the peaks for the three back axles are too small to be distinguishable from noise. The behavior in the previous pictures is also evident with cars, although they are usually lighter.

We also investigated the other reflectors as well as the second GBR in order to find the source of this high-frequency oscillation. While very distinct peaks, like in Figure 6.9, are also visible with other reflectors, reflector 3 shows the best results overall. No axle information could be extracted from the second GBR using our two methods, BANDPASS and WAVELET.

As described in Section 6.4.6, the hyperparameters for BANDPASS and WAVELET have been determined using the training data. To quantify the performance, Table 6.9 displays the results for all approaches. Again, MiniRocket has the best results for both categories. The MiniRocket confusion matrices are given in Figure 6.12. Especially for all vehicles, it achieves a BA of 76.7 %, delivering results comparable to Kawakatsu et al. [73], who achieve BA of 85.8 % using a single strain sensor. Admittedly, they do not exlude trailers beforehand. When excluding the cars, however, the BA decreases to only 69.4 %. BANDPASS and WAVELET have similar results for all vehicles, with WAVELET having a slightly better BA. Still, its BA is only 29.7 %. Both approaches tend to simply predict 2 axles. They improve for only

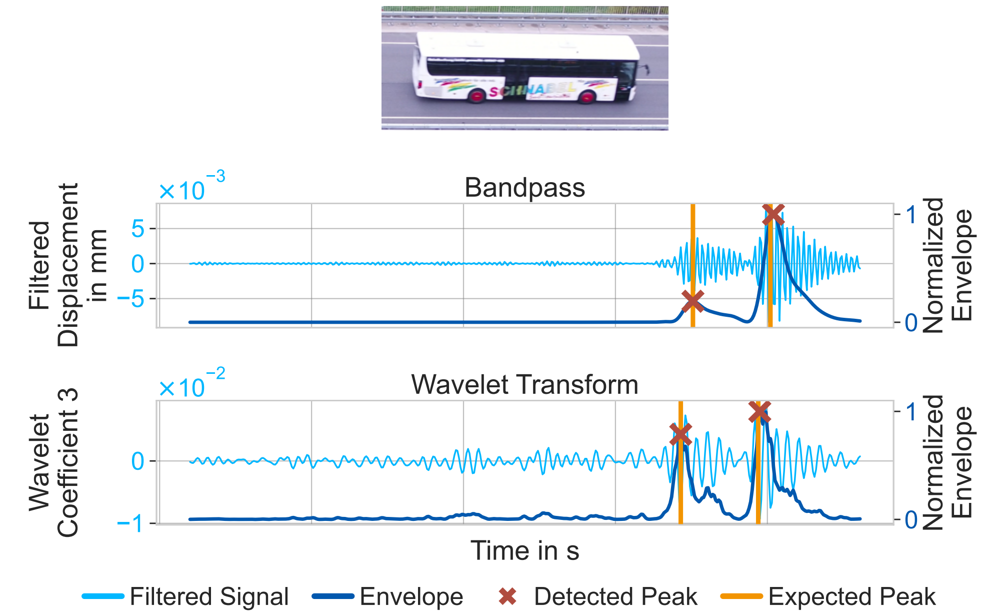


Fig. 6.9.: Example result of a bus for our filtering approaches. Adapted from [8] by changing the color palette.

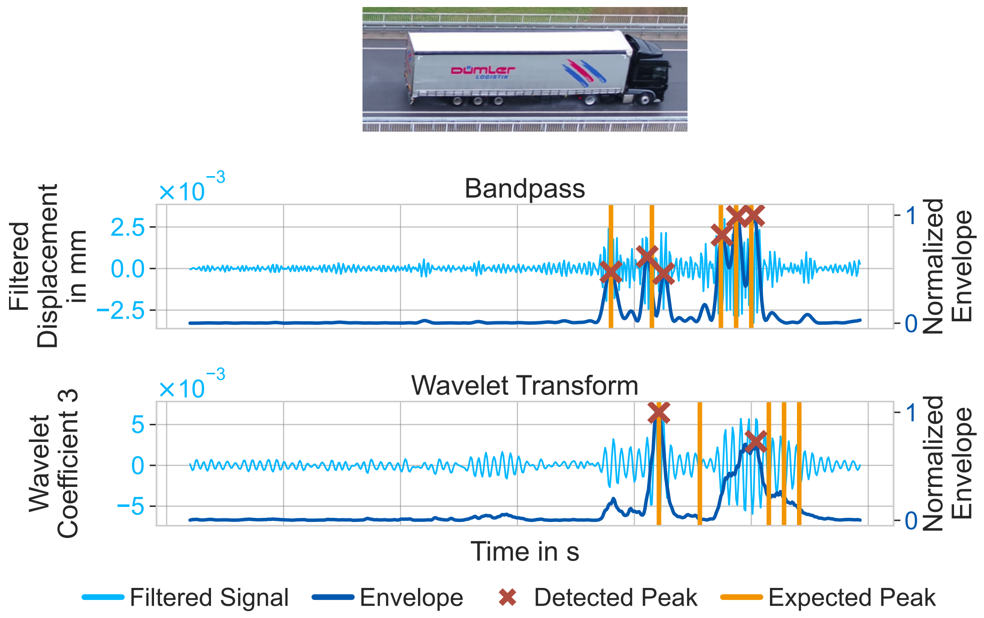


Fig. 6.10.: Example result of a truck for our filtering approaches. Adapted from [8] by changing the color palette.

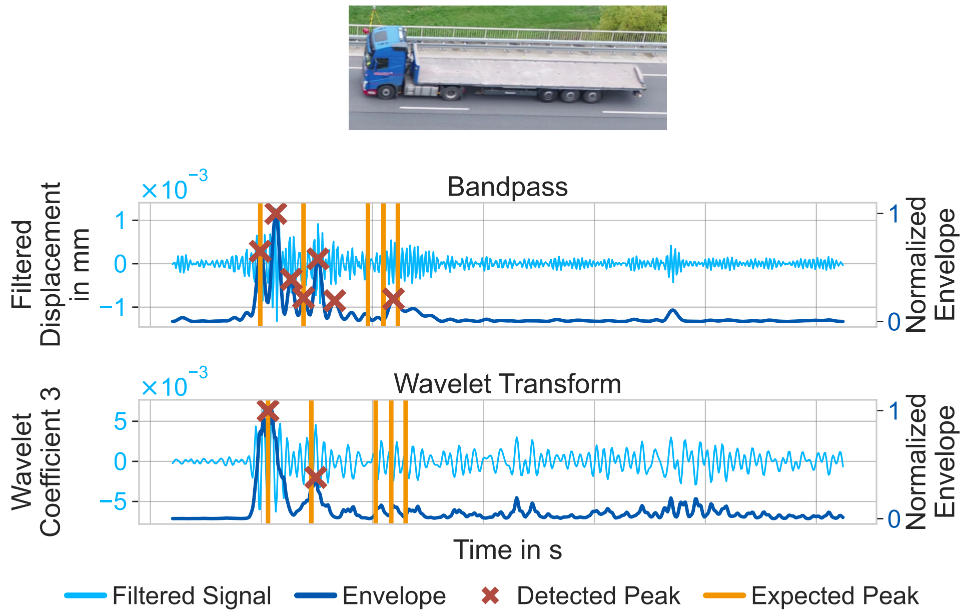


Fig. 6.11.: Second example result of a truck for our filtering approaches. Adapted from [8] by changing the color palette.

trucks, especially WAVELET, which reaches a BA of 57.5 %. RF, GB, and KNN tend to outperform BANDPASS and WAVELET but still fall heavily behind MiniRocket.

Tab. 6.9.: Classification results for axle count estimation with reflector 3 of Bridge A. The highlighted figures represent the best results for each vehicle group.

Model	**All Vehicles**				**Trucks Only**			
	BA **in %**	OA **in %**	P **in %**	RC **in %**	BA **in %**	OA **in %**	P **in %**	RC **in %**
BANDPASS	24.6	33.9	70.1	33.9	39.4	45.0	73.9	45.0
WAVELET	29.7	26.6	69.2	26.6	57.5	50.0	57.8	50.0
MiniRocket	**76.7**	94.5	94.5	94.5	**69.4**	75.0	75.5	75.0
RF	49.6	93.6	90.9	93.6	50.0	65.0	42.3	65.0
GB	46.7	92.7	91.8	92.7	61.3	65.0	63.2	65.0
KNN	46.7	92.7	91.8	92.7	61.3	65.0	64.1	65.0

6.5.6 Axle Spacing

As shown in the previous section, it is challenging to formalize a procedure for BANDPASS and WAVELET, which works for all vehicles. The same goes for axle spacing since it builds upon the peak detection. Yet, we want to indicate that the

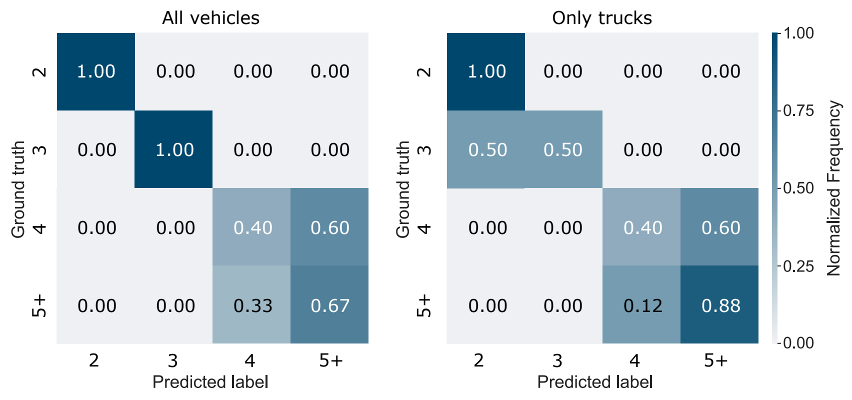


Fig. 6.12.: Confusion matrices for MiniRocket on the axle counting task for both vehicle groups.

information can sometimes be extracted successfully. A more generalized analysis will be performed afterward.

In Figure 6.9, two peaks are easily recognizable. Using their distance, the GBR sampling frequency of $200\,\mathrm{Hz}$, and the vehicle speed, we can calculate an axle spacing of $6.33\,\mathrm{m}$ using Equation (6.1). Comparing this to the UAV axle spacing of $6.19\,\mathrm{m}$, we overestimate the spacing by only $0.14\,\mathrm{m}$. Likewise, the results for Figure 6.10 would deliver close values, as the expected peaks are near the detected positions. Yet, additional peaks, such as the third peak in Figure 6.10 for BANDPASS, would lead to underestimating the second axle spacing, as we would also overestimate the axle count by one. These challenges are reflected in the overall results in Table 6.10, in which we compare the predicted axles spacings with the values extracted from the UAV image data. RMSE and MAE are extremely high and the R^2 is even harmful for both BANDPASS and WAVELET. As described Section 3.2.4, large deviations of the predictions from the reference values can lead to a negative R^2.
Harnessing cars and trucks, GB outperforms all other models with a MAE of $0.28\,\mathrm{m}$ and a R^2 of 0.83. MiniRocket follows close behind. KNN only achieves a R^2 of 0.59 with all vehicles, but after dropping cars it dominates its contenders reaching a R^2 of 0.91. Again, MiniRocket comes close behind as second, whereas GBs R^2 decreases. RMSE and MSE behave accordingly, with GB and KNN achieving the best MSE.

Using GB as the best performing model, we illustrate the several predictions results via a scatter plot in Figure 6.13. The identity line is included for orentation. Looking at the spacing between axle one and two, most values lie close to the optimal line. For higher values, however, the variance in prediction increases. More corncerning are the prediction for rearward axles. Many trucks have axle distances with a reference value of approximately $1.35\,\mathrm{m}$, but the prediction ranges from $0.2\,\mathrm{m}$ to $2.6\,\mathrm{m}$.

Tab. 6.10.: Regression results for axle spacing estimation with reflector 3 of Bridge A. The highlighted figures represent the best results for each vehicle group.

Model	**All Vehicles**			**Trucks Only**		
	R^2	RMSE in m	MAE in m	R^2	RMSE in m	MAE in m
BANDPASS	–9.31	3.76	2.45	–2.06	3.55	2.74
WAVELET	–9.26	3.75	2.53	–2.09	3.55	2.70
MiniRocket	0.79	0.54	0.30	0.87	0.74	0.49
RF	0.70	0.64	0.37	0.65	1.20	0.66
GB	0.83	0.49	**0.28**	0.79	0.91	0.54
KNN	0.59	0.75	0.37	0.91	0.60	**0.36**

6.6 Discussion

The central focus of this section is the potential of GBR displacement signals for a remote and data-driven BWIM. This section will discuss the results stated in the previous section. First, we will analyze the potential of ML for all classification and regression tasks. Afterwards, our filter approaches regarding axle configuration identification are discussed.

6.6.1 Machine Learning for Displacement-Based BWIM

As the results for vehicle-type classification are immaculate for feature-based methods, it is possible only to regard trucks for SHM if desired. Trucks cause a significantly greater deflection than cars. Therefore, the promising results are unsurprising. The issue of the driving lane can be avoided by using one reflector per lane as input data. Our distinction between vehicle types during vehicle configuration tasks is thus well founded.

It seems that, for all tasks except axle spacing estimation, ML models can extract vehicle configurations. MiniRocket, which uses the raw time series data and extracts features via convolutional kernels, shows especially auspicious results. The model with the best performance can vary from task to task, but MiniRocket either has the best performance or follows close behind. The overall satisfying results of our models indicate that the extracted configurations are identifiable from global bridge displacement data using data-driven ML approaches.

To acquire the vertical position of a vehicle, we test both a lane classification and a locus regression. The lane can successfully be extracted for both vehicle categories. Trucks can be classified almost perfectly, with only one vehicle misclassified. This

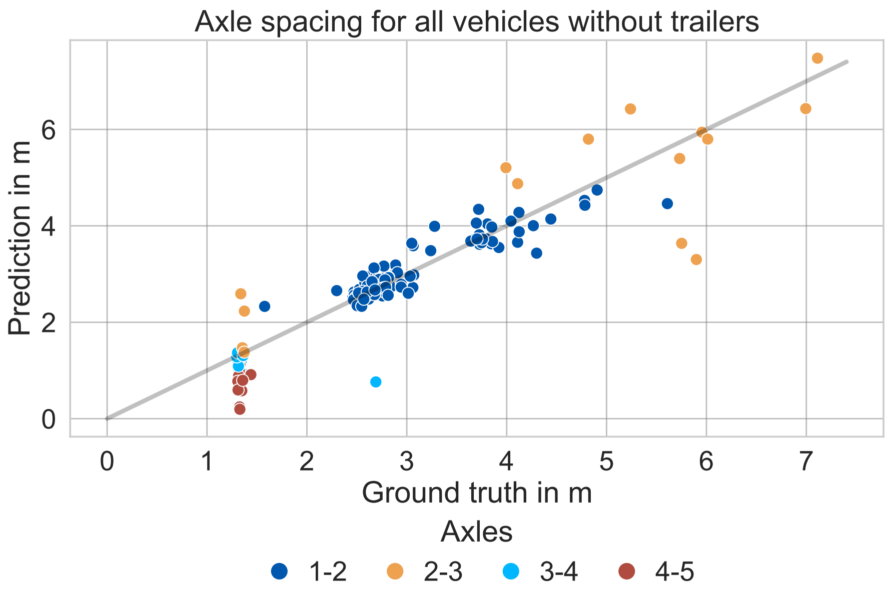


Fig. 6.13.: Scatterplot of the axle spacing estimation. The coloring of data points indicates the axle distance position. The identity line is included for orientation.

is unsurprising, as our reflectors are spread along the y axis (see Section 2.2). Accordingly, the maximum displacement during an event correlates to the driving side. Especially for heavy vehicles, a clear distinction can be made. In Figure 2.3, e.g., reflectors 3 to 5 show a steeper curve than reflector 1 and 2. This suggests that the vehicle drives in lane 1, which corresponds to the bus from Figure 6.9, which caused the bending. Thus, it seems enough for trucks to compare the signal power of reflectors 2 and 4 to acquire the lane.

The locus regression shows a similar behavior in that the results improve when discarding cars. Due to KNN's outstanding performance, we surmise that scaling our features will help with this task. This makes sense, as the maxima ratio within one event is more relevant than their absolute values. Investigating GB as an equally good model with SHAP confirms that one reflector per lane is necessary due to the different driving sides. In contrast to our lane selection of reflector 2, GB prefers reflector 1. This knowledge can be exploited in future investigations.

There is a significant difference in speed estimation results between all vehicles and only trucks. The reason for this might lie in the dataset composition. As cars are more frequent and with less variation in the vehicle configurations, like the length, the correlation between event duration and speed is more prominent. Thus, for some models, the R^2 decreases while the MAE improves. For MiniRocket, R^2 and

MAE decrease when discarding cars, suggesting that it does not mainly look at the event duration.

Although the results show that the detection of trailers is possible, the distinction is not as flawless as for vehicle type classification. MiniRocket as the best model only achieves a BA of 82.4 %. Then again, the task is more challenging than vehicle type classification. Interestingly, including cars does not improve the results although they heavily contribute to the no-trailer class. On the other hand, this might not help for the majority of truck trailers. Nevertheless, using only events without trailers for axle-related tasks seems appropriate.

The results for axle count classification imply that our features from Table 6.3 are not helpful for this task, as all models that depend upon them have a BA of less than 51 % on all vehicles. Only MiniRocket achieves satisfactory results, especially for all vehicles. The high OA but low BA show that the imbalance in the dataset, when including cars, leads to more misclassified trucks.

Comparing both datasets, it can be said that, although having more data generally helps, cars drastically change the distribution of many configurations. This makes it more challenging for models to learn truck properties. Regarding BWIM, where only trucks are considered relevant, this can be seen as a disadvantage. Since trucks come in significant axle count and spacing variations, our models naturally have generalization difficulties. With the aim of preventing too much variation, we factor events including trailers out. In case of heavy-duty trailers, this step looses us important SHM data points. However, the results prove us right. While the axle spacing estimation task is still challenging, we can achieve a MAE of $0.28\,\mathrm{m}$ for all vehicles and $0.36\,\mathrm{m}$ on trucks only. Figure 6.13 confirms these promising results. The predictions for the spacing between the first two axles lie especially close to the identity line. This makes sense, as this spacing has a smaller range due to many cars in the dataset and it is also present in every data point. The fact that the best model has difficulties with more rearward axle distances indicates that a more extensive dataset is necessary. Moreover, more complex models such as CNNs might be more useful for axle-related tasks. Especially, with respect to our filtering results. Then again, they also require more data. A workaround would be to focus on axle groups and not single axles. However, as there seems to be no exact definition for axle groups we abstain from such a task.

6.6.2 Filtering for Axle Configuration Identification

Figures 6.9 to 6.11 indicate that the information of axles is present in a bridge's global displacement time series. Individual examples produce promising results concerning axle count and axle spacing. To our knowledge, this has not been presented before, as displacement is barely exploited in BWIM. Compared to our ML approaches, BANDPASS and WAVELET have the advantage of providing explainable and localizable results. However, the results for BANDPASS and WAVELET in Table 6.9 and Table 6.10 show that it is challenging to find a general procedure. This is made clear by the negative R^2 values in Table 6.10. These values come as no surprise, since we showed on the basis of three examples that some axles are missed using BANDPASS and WAVELET and sometimes axles are falsely detected, leading to a large axles spacing error, especially for trucks.
Neither BANDPASS nor WAVELET can be described as the better approach overall, as their respective performances varied heavily between samples. Unfortunately, we can only show a small portion of all the recorded vehicles, as there are cases in which no axle-induced peaks are visible in the bandpass signal, but they exist with WAVELET.

While having a high SNR to measure these small vibrations is necessary, a good SNR alone will not lead to valuable measurements concerning axle detection. Interestingly enough, the weight of a vehicle or axle seems not to be the decisive factor, as for some cars, the peaks are easily recognizable, whereas they are not visible for trucks. Therefore, we assume that the relative axle weight within one event is relevant. The truck in Figure 6.11 appears heavier in the front than in the back. Accordingly, the peaks for the latter axles are less visible. However, this is only a conjecture since no axle load data have been recorded during this study. Also, the driving direction appears unrelated to how well axles can be detected. More peaks might have been detected with a higher sampling frequency, or at least their distinction might have been more straightforward for CWT. Yu et al. [157] have demonstrated that a sampling frequency of only 200 Hz leads to large errors. These vibrations seem to be caused at the junction between the bridge and the street. Consequently, this behavior might be specific to this bridge. Since we only monitored one field, we cannot say if the same vibrations can be measured for the other junction. Both fields seem too loosely coupled to transmit the signal if it exists. For one-span bridges, both leaving and entering might be observable. In such cases, our BANDPASS and WAVELET approaches can also extract the vehicle speed.

For vehicles with clearly visible peaks, like in Figure 6.9, the axle-induced response could also be observed in the signal of other reflectors. This suggests that the specific reflector or its attachment does not induce the vibration but is a high-frequency vibration in the bridge displacement. Also, as mentioned in Section 2.2, we monitor the bridge with two GBRs that measure different components. There have not been visible vibrations or peaks for the GBR measuring the z and y components, although the SNR lies in an adequate range. This indicates that the vibration mainly occurs in the x-component. However, a more detailed investigation is necessary.

6.7 Conclusion and Outlook

In this chapter, we discuss a novel data-driven, displacement-based BWIM approach. The data was recorded at a two-span bridge in Germany using GBR and a UAV for reference data. We investigate the potential of both classic signal processing and ML to extract the vehicle configurations from bridge-crossing events. These configurations include vehicle type, speed, lane, locus, axle count, and spacing. One challenge herein is that displacement is a non-local bridge response. Furthermore, our dataset is imbalanced and consists of variable-length time series. We evaluate all approaches on all vehicles and all trucks only.

As ML approaches, we test four different models, three of which depend on manually crafted features. The fourth model, MiniRocket, uses the raw time series data. Over all configurations, MiniRocket achieves the most auspicious results, comparable to other studies such as Kawakatsu et al. [73]. While speed, trailer-presence, lane and locus can be extracted, axle count classification is more challenging for all models. Only MiniRocket can classify a truck axle count with a BA of $76.7\,\%$. Finally, the results for axle spacing show promising results wih a MAE of only $0.28\,\mathrm{m}$ for all vehicles. That said, we exclude events including trailers to reduce complexity for axle count and axle spacing estimation. These events show a great variation in configuration but are barely represented in the dataset. Therefore, more measurement campaigns need to be carried out. In addition, more complex models and using extracted values like speed as an input feature could also lead to better results.

We have recorded a high-frequency vibration for this bridge that coincides with axles crossing the junction. Using bandpass filtering and wavelet transform, we could demonstrate examples of this behavior. Based on this finding, we try two approaches for axle count and axle spacing determination purely based on signal processing. While the information seems available, we have not found a comprehensive procedure for the automatic extraction of axle configurations. A more sophisticated approach exploiting CNNs might be more successful as they can learn more generalizable features. For this, however, either the dataset needs to be increased, or data augmentation needs to be applied, as it has been investigated in Chapter 5. Furthermore, it must be investigated whether other bridges show a similar behavior in the high-frequency range. Ideally, GBRs with a higher sampling rate will be used for these measurements.

We showed that a purely data-driven BWIM exploiting GBR-displacement time series data is possible. However, these promising results must be refined using a more extensive dataset. Ojio et al. [115] indicated that determining axle loads is possible with bridge displacement signals and the vehicle configurations extracted in this study. Thus, our results work towards a fully remote and data-driven BWIM and fulfill our **Research Goal II**.

Part III

Synopsis

7 Methods for data-driven detection and classification of vehicles in non-invasive radar data (MONITOR)

You can walk as far as you want, everywhere you go you only see your own horizon.

— **Max Eyth**
(Engineer, Author)

In this final chapter, we will investigate (1) all developed methods in a single pipeline and (2) how our models work on new measurements after training them on previous campaigns. Thus, this chapter introduces our final pipeline MONITOR (see Figure 7.2) and simulates how it would perform in a real-life usecase. To this end, we retrain the best models of Chapter 5 on the first four measurement campaigns of Bridge A and test them on the last one. For the event detection, we will use the CNN developed in Sections 4.5 and 4.6. It will not be retrained as no data of the fifth campaign was used during its development. Extracted events are then passed into the RF of Section 5.5 trained without data augmentation but with Bridge A and Bridge B data. However, we will only regard single vs. multi event classification by merging all classes with more than one vehicle. Exploiting single events, we extract the vehicle type and locus using GB, and lane, speed and trailer-presence using MiniRocket (see Chapter 6). For vehicles without trailers, MiniRocket is used to estimate the axle count and GB for axle spacing estimation.

In Section 7.1, we will shortly explain the dataset and split used for this task. Section 7.2 gives insight to the results and offers an extensive discussion. Finally, in Section 7.3 a conclusion with regards to our MONITOR is given.

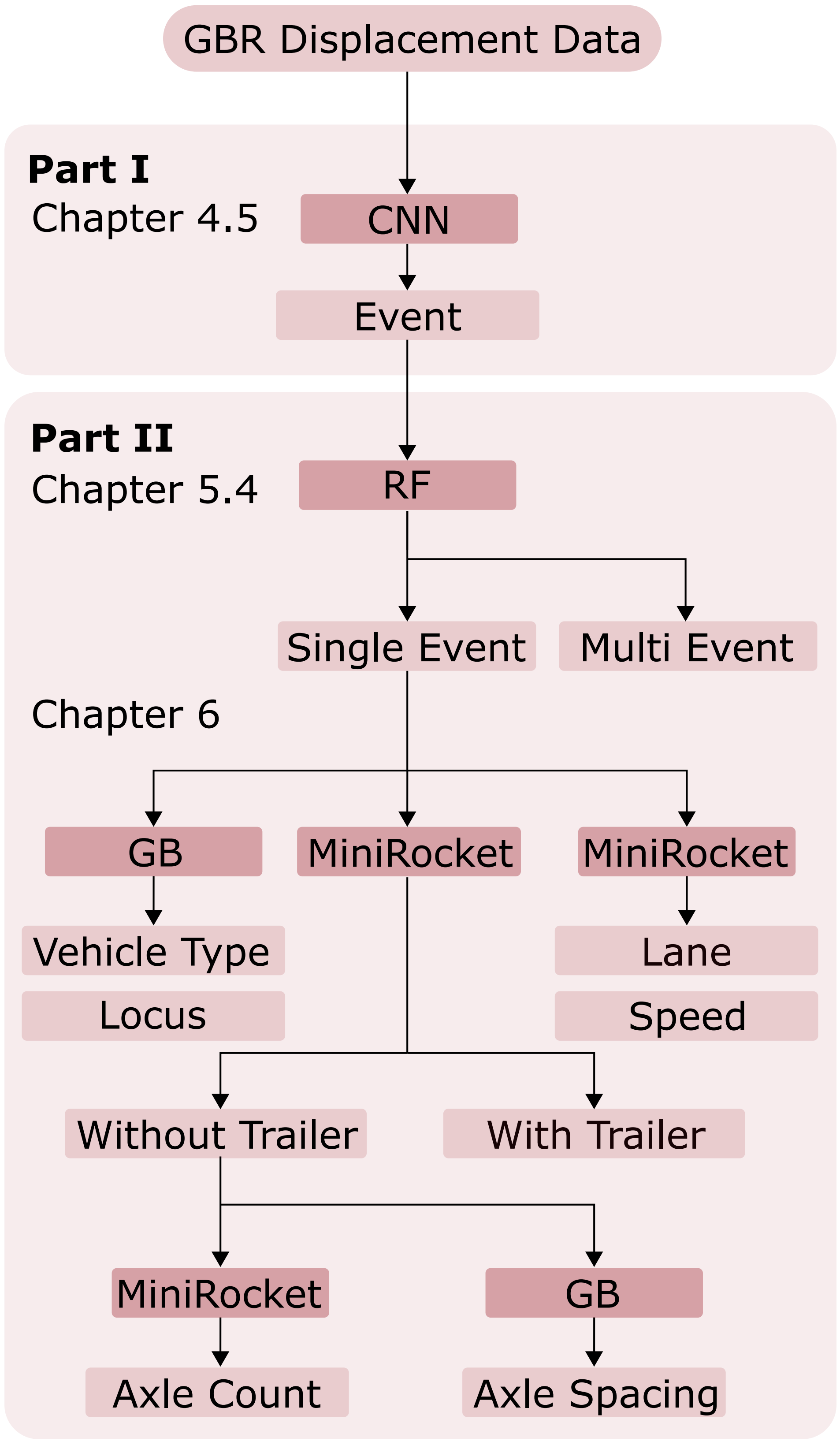


Fig. 7.1.: Overview of the pipeline MONITOR.

7.1 Dataset

As mentioned, we train all models on the first four measurement campaigns and test on the final one. This procedure simulates a real-world usecase. In such a scenario, we would repeatedly monitor a bridge both with GBR and a UAV in order to collect reference data. After training models on the acquired dataset, the UAV monitoring would be omitted in following measurement campaigns and only the GBR data would be used to detect and categorize traffic. When training the RF for vehicle count estimation, we also include the first Bridge B dataset as per Section 5.5. All measurement campaigns and the UAV recorded events are listed in Table 7.1.

Tab. 7.1.: Overview of Bridge A and Bridge B UAV measurement campaigns used in this chapter.

Bridge	Date	Events	Set
B	22.10.2019	354	Training
	23.10.2019	174	Training
	24.10.2019	73	Training
A	27.02.2020	380	Training
	29.07.2020	288	Training
	09.06.2021	47	Test

Figure 7.2 presents the time series for the UAV assisted part of the final campaign of Bridge A. That campaign had to be cut short due to a thunderstorm preventing further UAV flights. The negative offset stems from phase jumps caused e.g. by people walking through the GBR LOS. Overall, 47 events have been registered during the time span of $662\,\mathrm{s}$, whereof 4 are multi events.

7.2 Results and Discussion of the applied MONITOR pipeline

In this section, we will iteratively go through all steps of MONITOR, starting with the event detection. In all steps, we proceed with the evaluation in the same way as in the corresponding parts of this thesis. In the case of vehicle classification or regression tasks, first consider the ideal scenario, i.e. we take the test data directly from the database. This is done with the intention of showing that the model generally works and to acquire a baseline. For reason of comparability, the baseline is limited to events which are still in the MONITOR pool at the corresponding step. Then, we apply the model to the events extracted with our CNN. If possible, SHAP

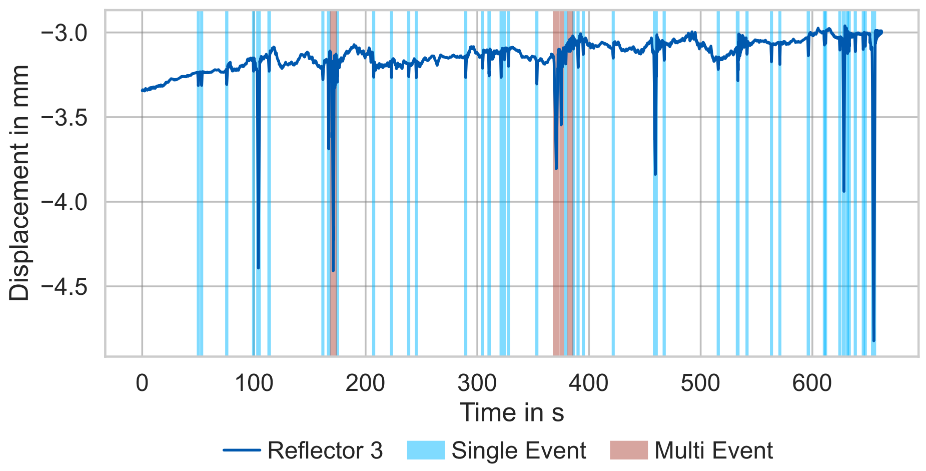


Fig. 7.2.: UAV-labeled time series for the measurement campaign from 06.09.2021. Single and multi events are highlighted seperately.

values are used to grasp details of the decision-making process. Results which would lead to nonsensical results in subsequent steps are kept track off, however, they are discarded for the following steps. For instance, multi events classified as single events are not evaluated in single event classification tasks but counted as false results. Finally, we will showcase some examples again resorting to SHAP analysis.

7.2.1 Event Detection

As mentioned, we use the CNN from Section 4.5 to detect events. Our approach of segmenting the time series to extract events equates the one presented in Section 4.6. This results in the confusion matrix displayed in Figure 7.3 and the segmentations highlighted in Figure 7.4. 43 events are detected succesfully, four are missed and at the same time two non-events are detected erroneously. Again, "missed" is defined as not capturing the center of an event. Two events are not detected at all. In two other cases, the window only captures the lowest point of an crossing, incorrectly identfying it as non-event. As the slopes are detected, those two events are divided in two parts accounting for the FN. In other words, a disadvantage of this approach is that it is very dependent on the viewport and thus the starting point. An unfortunate starting point could lead to a frequent alignment of the CNN window with the lowest point of an event. Consequently, many events would be split in two, due to the misclassification of such windows. A threshold does not suffer from this specific shortcoming. Another option would be to retain memory in future model architectures or to train on sliding windows with a bigger viewport.

In one case, two single events are falsely merged to a multi event. Although they might be correctly classified as multi events they are not considered for vehicle count estimation, leaving us with 41 events (see Table 7.2).

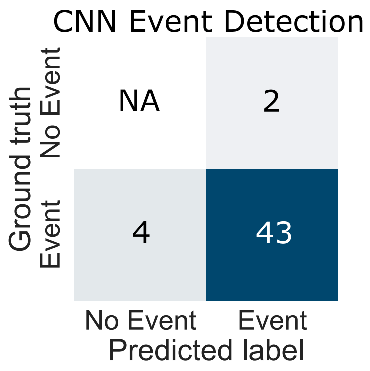


Fig. 7.3.: Confusion matrix for the event detection results.

Tab. 7.2.: Event detection results.

Events	Detected	Incorrect Detections	Remaining
47	43	2	41

Briefly looking at how precise the start time and end time can be located, we again use the MAE and RMSE stated in Table 7.3. The values lie close to the ones from Section 4.6. In fact, they are superior but also stem from fewer data points. The majority of events get overestimated in duration, as desired, but sometimes a part of an event is missed.

Tab. 7.3.: Results of the CNN for the Bridge A time series displayed in Figure 7.4. It is evaluated on MAE and RMSE on the detected the start time and end time.

Parameter	MAE in s	RMSE in s
Start time	0.17	0.19
End time	0.13	0.16

7.2.2 Vehicle Count

We use the RF of Section 5.5 to differentiate between single and multi events. This is achieved by grouping classes with more than one vehicle together to multi events. For the purpose of consistency, we train this specific model on both Bridge A and Bridge B data. When testing on the UAV data the left confusion matrix in Figure 7.5

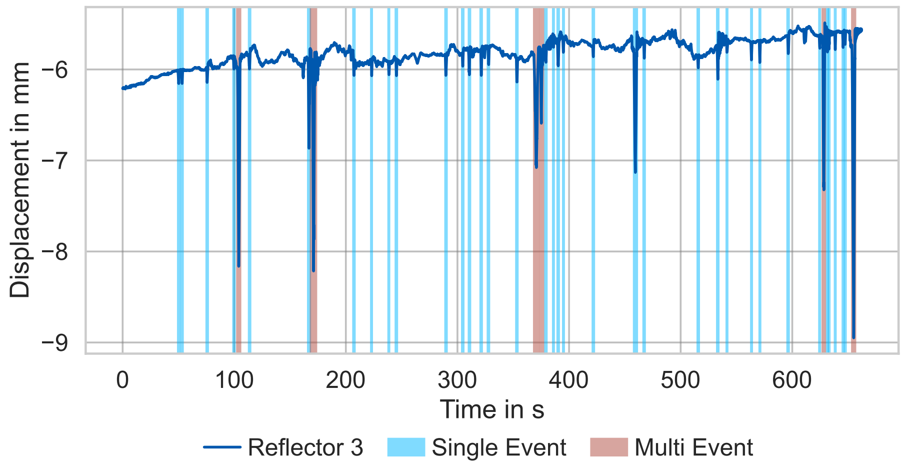


Fig. 7.4.: UAV-labeled time series for the measurement campaign from 06.09.2021. Detected single and multi events are highlighted.

is produced. We evaluate the baseline on the same 41 succesfully detected events MONITOR retained until this point. It shows that for this measurement campaign only one event is misclassified when having precise start and end points.
Applying the same model to the events extracted via our pipeline leads to worse results, since now three single event are falsely labeled as multi events. Unfortunately, these three are also single truck events. Keeping in mind that trucks are especially relevant for SHM, we lose three important data points. On a more positive note, all multi events are matched correctly. Thus, we would not overestimate the load by attributing the load of several vehicles to one individual vehicle.

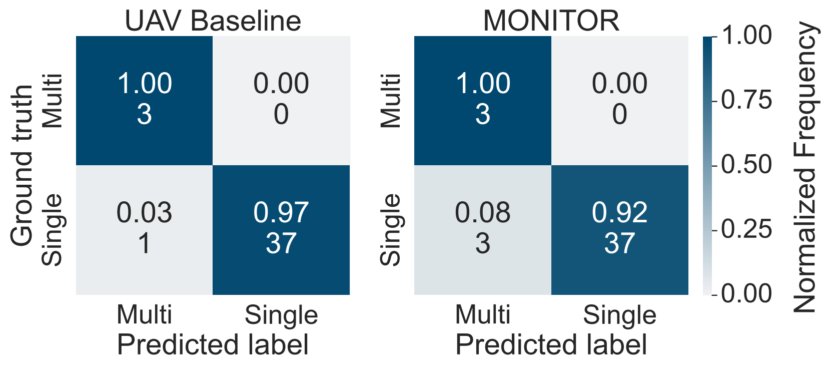


Fig. 7.5.: Confusion matrices for single vs. multi event classification on the UAV database (left) and the detected events (right). Both normalized and absolute values are given.

In an attempt to understand these differences we survey the waterfall plot (see Section 3.2.5) for all three misclassifications. The left column in Figure 7.6 shows the SHAP values when using the UAV dataset and on the right detected events are depicted. Plots in the same row stem from the same vehicle. Special attention must be paid to the abolute value of the "Length" feature. Its impact increases drastically with a longer event duration. In the first two examples, the event detection overestimates the event duration by more than $0.5\,\mathrm{s}$. In turn, the SHAP values almost triple in size for both "Length0" and "Length1", easily pushing the prediction towards multi event. An equivalent conclusion can be drawn from the second example. With approximately $1.85\,\mathrm{s}$ the UAV duration is so small that the "Length" feature is not even listed among the most important 7 features. The corresponding detected event, however, has a duration of $2.5\,\mathrm{s}$. Accordingly, "Length0" and "Length1" contribute primarily to a multi event prediction. Finally, the last row is a truck which is misclassified in both cases. Here, the event duration is captured almost perfectly with $3\,\mathrm{s}$ compared to $2.95\,\mathrm{s}$. However, since the crossing simply takes long, the "Length" feature is still predominant. In the end, what we aimed to avoid in Chapter 5 became true after all. We tried to reduce the importance of the signal length but since we also try to overestimate the event duration in order to capture all relevant information, we counteract our own efforts. Other features, such as the "Skewness" are also considerably influenced by not finding the perfect start and end point of an event. One solution appears to be to train the model without assuming perfect event segmentation. We have tried this and it lead to worse results for single vs. multi event classifcation. With the observations at hand further investigations in this regard need to be considered. Futhermore, a higher event segmentation resolution would improve our results. In conclusion, we now have 35 single vehicles left on the basis of which we can now evaluate the vehicle paramter estimation.

7.2.3 Vehicle Classification

In the final step of MONITOR, we try to extract vehicle parameters from single event crossings. When reasonable, we avoid using MiniRocket or KNN as we cannot analyze their predictions with SHAP. Again, we use the UAV dataset as a reference but only events which are also still present for MONITOR.

Vehicle Type

Figure 7.7 displays the vehicly type classification results. Expectedly the majority of vehicles are identfied correctly. In fact, the UAV baseline and MONITOR achieve

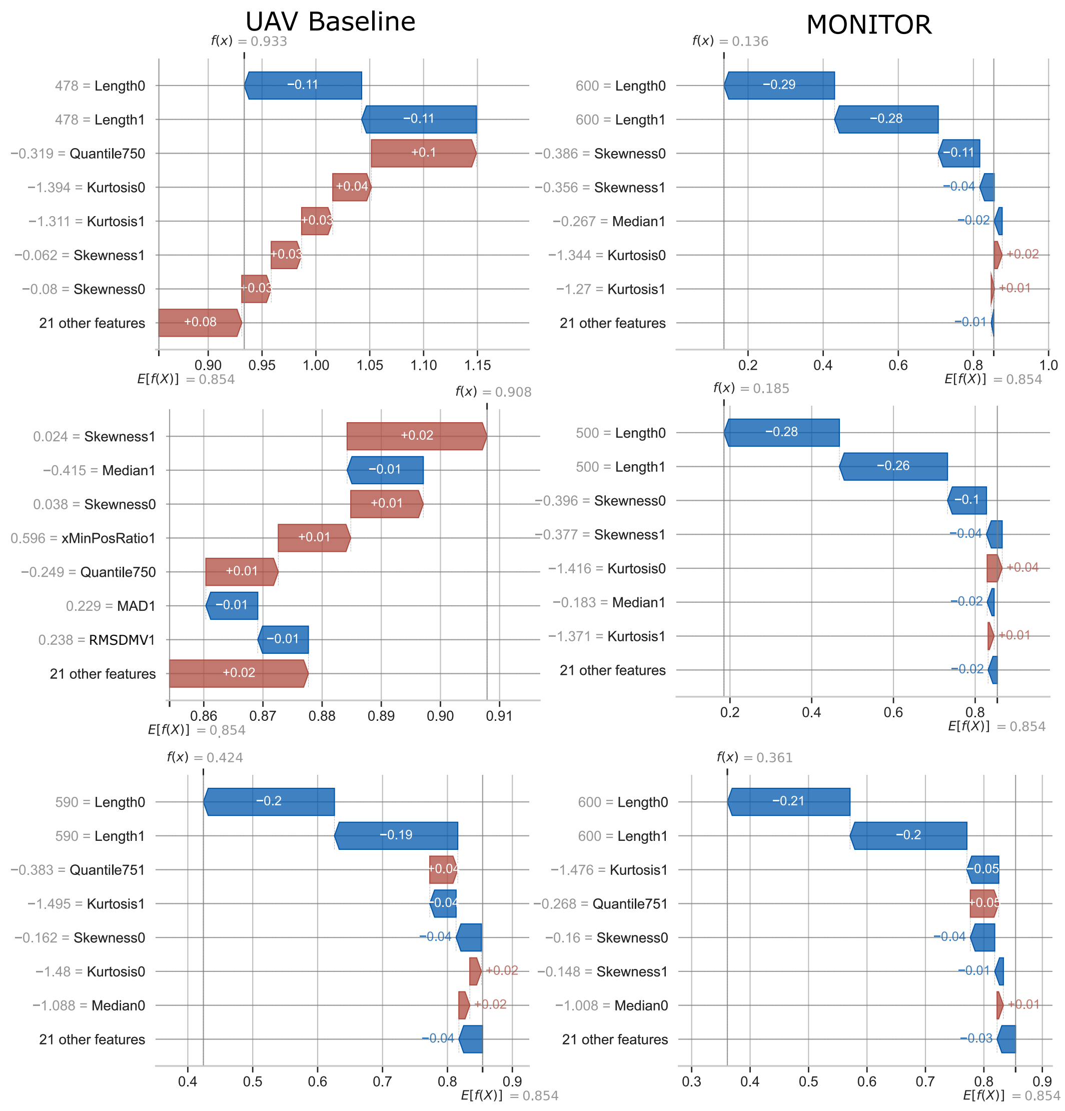


Fig. 7.6.: Waterfall plots for three different vehicles, one per row. In the left column the SHAP values for predicitons on UAV events. The right column contains the corresponding events extracted via our CNN.

identical results. The only misclassification is a smaller truck. Thus, this task seems to be robust against event length variations due to our segmentation.

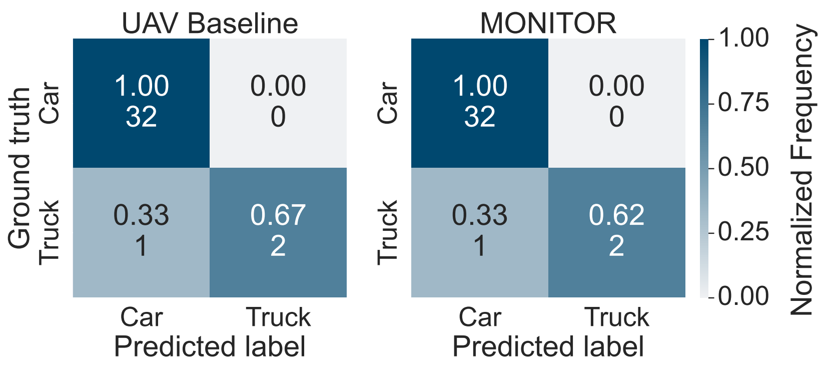


Fig. 7.7.: Confusion matrices for vehicle type classification on the UAV dataset (left) and the MONITOR results (right). Both normalized and absolute values are given.

Lane

Interestingly, MONITOR outperforms our UAV baseline when extracting lane information using MiniRocket. MONITOR performs immaculately, while the baseline has one FN. Thus, lane identification also seems to be resilient against imprecisions in the event segmentation.

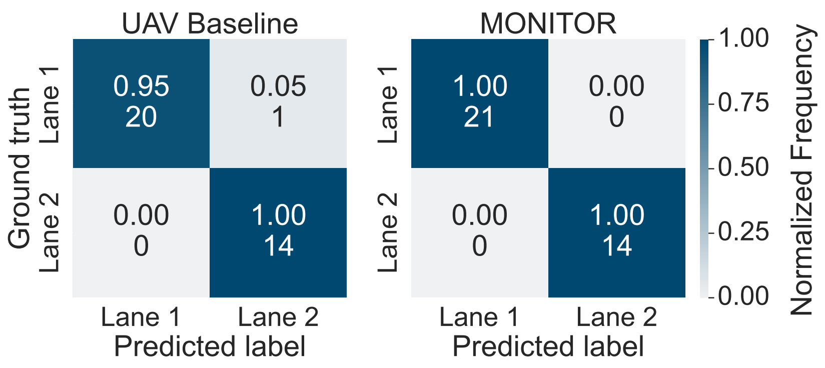


Fig. 7.8.: Confusion matrices for lane classification on the UAV dataset (left) and the detected events (right). Both normalized and absolute values are given.

Locus

A finer resolution in the lateral position is given by the locus. Table 7.4 states the results for locus estimation using GB. The results are comparable over both data sources, with UAV data being better overall. Still, despite the segmentation GB still achieves a R^2 of 0.85. Considering we lose three trucks during single event classification, we now mainly evaluate on cars. Thus, with more extrem locus values due to trucks the results might worsen.

Tab. 7.4.: Results for the locus estimation using GB.

Dataset	R^2	MAE in m	RMSE in m
UAV	0.90	0.48	0.59
MONITOR	0.85	0.59	0.70

Speed

Speed is the first vehicle parameter, in which MONITOR has considerably worse results than the UAV baseline. The MAE drops by $0.93\,\mathrm{m}$ and the R^2 by 0.16. Intuitively, this makes sense as the event duration is a relevant value for speed calculation and correlates highly with it. It seems MiniRocket also depends heavily on it. At least, we assume the variations in start and end point detection during MONITOR event segmentation is the reason for the differences in performance. Unfortunately, it is currently impossible to investigate MiniRocket with SHAP.

Tab. 7.5.: Results for the speed estimation using MiniRocket.

Dataset	R^2	MAE in m/s	RMSE in m/s
UAV	0.83	1.09	1.42
MONITOR	0.67	2.02	1.68

Trailer

In the interest of reducing the complexity for axle-related tasks, we have only investigated vehicles without trailers. Beforehand, we have shown that MiniRocket can detect trailers with some difficulties. The results within our pipeline are displayed in Figure 7.9. For the second time MONITOR surpasses the UAV baseline. With the baseline data MiniRocket misses one trailer. Unfortunately, we cannot use SHAP on MiniRocket to gain more insights into the decision process. Initially, we strongly suspected the difference between baseline and MONITOR to be rooted in the overestimation of the crossing duration by our segmentation. Events naturally take longer if a trailer is attached to a vehicle. Conversely, two cars with trailers have been classified correctly and at the same time no false trailers have been attached to slower trucks.

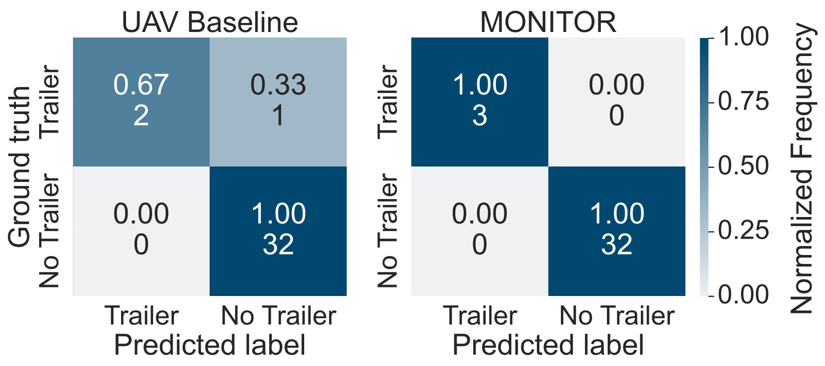


Fig. 7.9.: Confusion matrices for trailer detection on the UAV dataset (left) and with MONITOR (right). Both normalized and absolute values are given.

One example of those cars is illustrated in Figure 7.10. One indicator for trailer-presence might be that stronger bridge vibration oftentimes correlate with trailers. This is indeed the case for one of those two cars. Overall, the classification surprisingly achieves satisfactory results. After this distinction, we are left with 32 vehicles two of which are trucks.

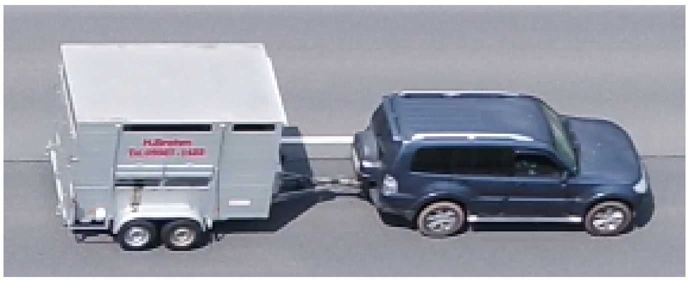


Fig. 7.10.: Car with trailer correctly identified by MONITOR.

Axle Count

With the elimination of trailers and the misclassifications of trucks in the single event detection step, we now are left with mainly two-axle vehicles. In fact, only one vehicle has more than two axles namely three. MiniRocket achieves the results stated in Figure 7.11. Unsurprisingly, two axle vehicles, are identified almost perfectly. In both cases only the three-axle vehicle is predicted incorrectly. The baseline sees it as a two-axle vehicle while MONITOR counts four axles.

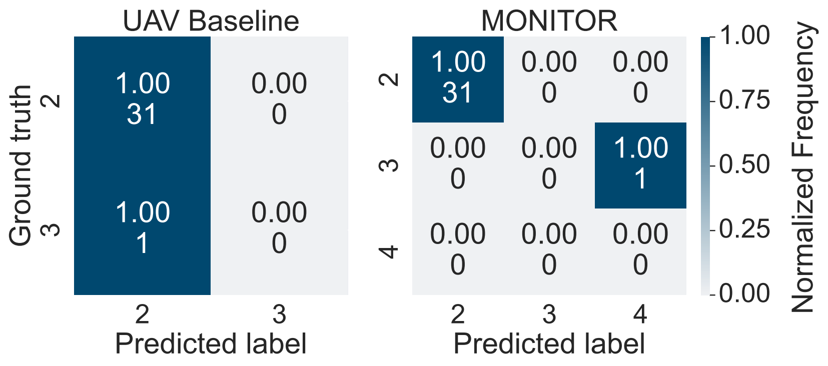


Fig. 7.11.: Confusion matrices for axle count classification on the UAV dataset (left) with MONITOR (right). Both normalized and absolute values are given.

Axle Spacing

Finally, we try to estimate axle spacing via GB from trailer-free events. The results are given in Table 7.6. When testing on the UAV baseline, the GB performs acceptable. MONITOR results drop extremly in R^2. Still, the MAE is only $0.33\,\mathrm{m}$. The RMSE more than doubles compared to the baseline. Just like speed, axle spacing estimation seems to be very dependent on the signal length. The superior UAV results serve as yet another motivation to improve the MONITOR event detection.

Tab. 7.6.: Results for the axle spacing estimation using GB.

Dataset	R^2	MAE in m	RMSE in m
UAV	0.68	0.23	0.30
MONITOR	–0.62	0.33	0.69

7.2.4 Examples

After analyzing all vehicle parameters, we now want to share four examples for further insights. They include one car and three trucks whereof one has a trailer. Each example includes an image of the vehicle, a table with the MONITOR results and the reference values. Axle count and spacing are not estimated for vehicles with trailers.

Example 1: Car

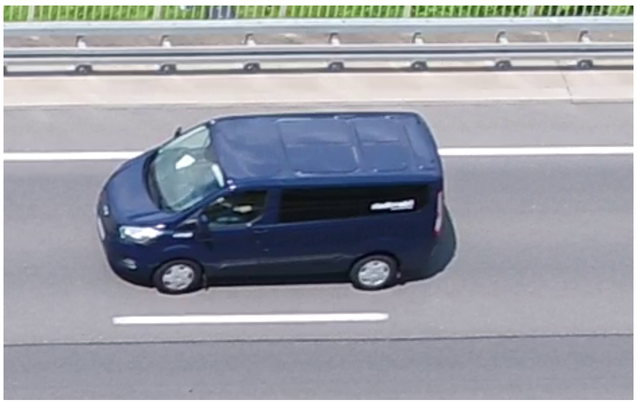


Fig. 7.12.: Image of the Example 1 car, corresponding to the results in Table 7.7.

The first example is the crossing of an Ford Transit Custom, which we drove ourselves. Our version has an axle spacing of $2.93\,\mathrm{m}$. Its example is the representative of the common car. Lane, vehicle type and number of axles are recognized correctly. Furthermore, locus and axle spacing estimations lie close to the UAV reference. The axle spacing reference itself only has an error of $0.03\,\mathrm{m}$ compared to the vehicle specifications. Only speed has a considerable disagreement, overestimating the vaule by $3.38\,\mathrm{m\,s^{-1}}$.

Tab. 7.7.: Table for Example 1 car depicted in Figure 7.12.

Parameters	MONITOR	UAV Reference
Vehicle Type	Car	Car
Lane	2	2
Locus	$6.86\,\mathrm{m}$	$6.48\,\mathrm{m}$
Speed	$21.98\,\mathrm{m\,s^{-1}}$	$18.61\,\mathrm{m\,s^{-1}}$
Axle Count	2	2
Axle Spacing	$2.64\,\mathrm{m}$	$2.89\,\mathrm{m}$

Example 2: Light Truck

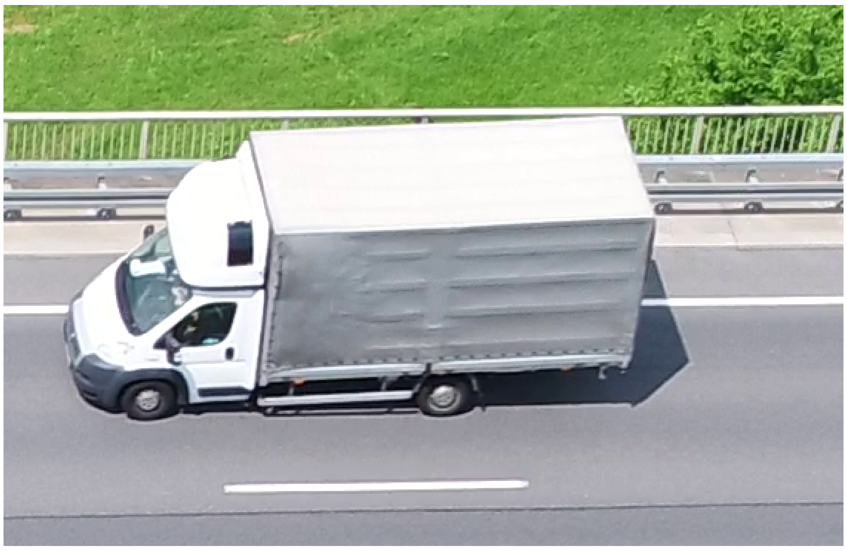


Fig. 7.13.: Image of the Example 2 truck corresponding to the results in Table 7.8

This example showcases the only misclassification for vehicle type with MONITOR. This is an edge case, as some might not refer to such a vehicle as truck. It almost certainly is a light-weight truck, which makes the prediction more comprehensible as the load certainly plays a big part in it. The lane and number of axles are assessed correctly, and the locus estimation achieves satisfying results. Compared to the previous car, the determined speed lies closer to the reference value. However, the axle spacing is underestimated by almost $0.5\,\mathrm{m}$.

Tab. 7.8.: Table for the Example 2 truck depicted in Figure 7.13.

Parameters	MONITOR	**Reference**
Vehicle Type	Car	Truck
Lane	2	2
Locus	$6.29\,\mathrm{m}$	$6.05\,\mathrm{m}$
Speed	$24.15\,\mathrm{m\,s^{-1}}$	$24.43\,\mathrm{m\,s^{-1}}$
Axle Count	2	2
Axle Spacing	$3.59\,\mathrm{m}$	$4.05\,\mathrm{m}$

Example 3: Truck

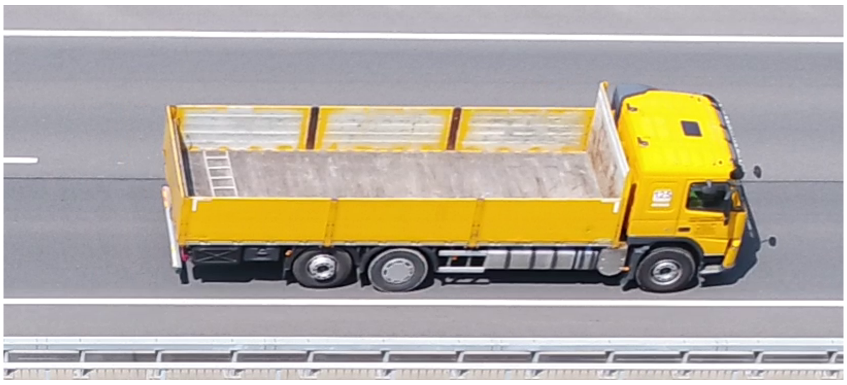


Fig. 7.14.: Image of the Example 3 truck corresponding to the results in Table 7.9

Figure 7.14 shows the only vehicle with more than two axles which reached the last step of our MONITOR test. Its image highlights some challenge in labeling the reference data. Arguably, the last axle could also be lifted from the ground, rendering this a two-axle vehicle. Especially, as the the truck is unloaded. This illustrates that even visually axle count estimation is challenging. Computer vision methods might fail at this task, particularly if the view is not as clear as in this image due to weather conditions.

Since we detect one axle too much, we list three axle spacings for MONITOR although only two exist. The first spacing lies somewhat close to its reference with an error of $0.83\,\mathrm{m}$. The second value, however, is extremly wrong. This fits our observations from Section 6.5.6, where a second axle distance of around $1.35\,\mathrm{m}$ can lead to very different predictions, although not to this extent. Apart from the challenges in axle-related tasks, other parameters achieve good results.

Tab. 7.9.: Table for the Example 3 truck depicted in Figure 7.14.

Parameters	MONITOR	Reference
Vehicle Type	Truck	Truck
Lane	1	1
Locus	$10.13\,\mathrm{m}$	$10.81\,\mathrm{m}$
Speed	$20.98\,\mathrm{m\,s^{-1}}$	$20.23\,\mathrm{m\,s^{-1}}$
Axle Count	4	3
Axle Spacing	[$4.08\,\mathrm{m}$, $4.78\,\mathrm{m}$, $1.26\,\mathrm{m}$]	[$4.91\,\mathrm{m}$, $1.37\,\mathrm{m}$]

Example 4: Truck with trailer

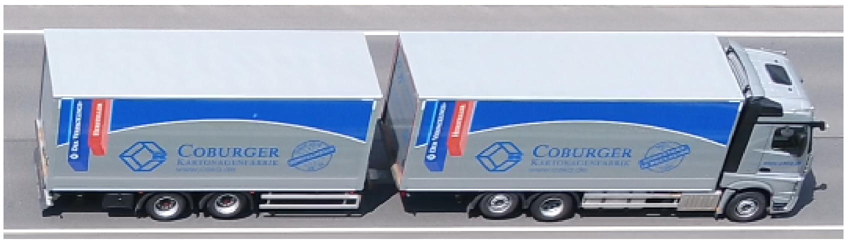


Fig. 7.15.: Image of the Example 4 truck corresponding to the results in Table 7.10

The final example constitutes a truck with a trailer (see Figure 7.15). Despite the length of the vehicle, the speed is estimated very accurately. The trailer in this example was detected correctly. Thus, axle count and spacing are not estimated. Again, however, two axles, namely the third and fourth are lifted, which is barely recognizable. Consequently, the distance between the second and third axle is very big. In order to not be forced to exclude such vehicles from axle count and spacing tasks, more data needs to be collected.

Tab. 7.10.: Table for the exmaple 4 truck depicted in Figure 7.15.

Parameters	MONITOR	**Reference**
Vehicle Type	Truck	Truck
Lane	1	1
Locus	10.34 m	11.05 m
Speed	$14.40\,\mathrm{m\,s^{-1}}$	$15.0\,\mathrm{m\,s^{-1}}$

7.3 Conclusion

In this chapter, we combine all previously designed and analyzed methods to one single pipeline called MONITOR. To simulate a real-world scenario, we train all models on the first four Bridge A measurement campaigns and evaluate on the remaining fifth one.

Our event detection works fine overall but we would have four non-sensical results due to FN classifications. Moreover, we miss two events. The greatest weakness of the event detection is the imprecision in segmenting events. Our goal during development has been to not miss any part of an event in order to not lose any bridge displacement information. Thus, we would rather overestimate the event duration. This can help with filtering approaches to detect axles as described in Section 6.5.5. On the other hand, our classification models have been trained on perfectly extracted displacement sequences and therefore have difficulties in such a scenario. This becomes evident during the single event detection step of MONITOR. Despite our greatest efforts in Chapter 5 to minimize the impact of the signal length feature on the classification of events according to the amount of vehices involved, three single trucks have been falsely assigned to the multi event class. SHAP illustrates that the event duration and to that effect the overestimation due to our event segmentation are the main factor here. Regarding SHM the loss of three out of five trucks weighs heavily.
More pleasingly are the results for vehicle type classification and lane detection as well as locus estimation. In particular the trailer detection surprised us as we expected misclassifications due to the event duration. On the contrary, all MONITOR classifications are flawless, even detecting cars with small trailers.
Finally, axle-related tasks suffer from the small dataset both in training and during evaluation. With mostly two-axle vehicles other cases can barely be investigated. Especially, as the only vehicle with three axles is misclassified by MONITOR. Nevertheless, two-axle vehicles are identified and the first axle spacing is estimated satisfactorily.

In sum, we show in this chapter that it is possible to develop a single pipeline for GBR-based, remote BWIM exploiting only data-driven methods. Despite some remaining challenges, this has never been achieved before and serves as a clearly defined basis for all future studies.

Conclusions and Outlook 8

This final chapter summarizes the significant contributions of this thesis and recapitulates the main objective and research goals, as well as the MONITOR. We discuss the potential and limitations of our approaches in Section 8.2 and propose opportunities for future studies in Section 8.2.

8.1 Concluding Summary

The main objective of this thesis is to develop methods that can extract traffic information and vehicle parameters from GBR bridge displacement data. Until now, no comparable study in this regard has been conducted. We established two research goals: The detection of bridge crossing events in the time series displacement data (Part I) and the analysis of events concerning the traffic causing the bridge displacement (Part II). Although we have separated both parts, all approaches have been designed so that they can be merged into a single pipeline called MONITOR. By applying it in a real-world scenario, we ultimately showcase that data-driven methods can extract vehicle information from raw bridge displacement data.

8.1.1 Conclusions for Part I

Part I assesses the detection of vehicle crossings in GBR bridge displacement time series data. We aim to achieve real-time capability, as we want to be able to detect overweight trucks during traffic control. To this end, we split the time series into $0.5\,\mathrm{s}$ long windows and predict the presence of a vehicle for each window individually. We successfully address the long-term drift caused by, e.g., environmental influences by removing the mean of each window. Predictions are then made by exploiting shallow learning and DL using the data from two bridges. The DL approach is, in the end, refined to a segmentation process.

Shallow Learning In the first study, we investigate to what extent various shallow learners can detect vehicles based on manually extracted features. Ultimately, events are recognized with an OA of up to $83.8\,\%$. A more in-depth analysis, however, shows that models have difficulties distinguishing actual crossing events from subsequent bridge oscillations. After a heavy vehicle leaves the bridge, the same would sometimes oscillate in its eigenfrequency for some time after. But as there is no vehicle on top of the bridge, oscillations are not part of an event.

Deep Learning DL in the shape of a CNN is evaluated for a second approach. Again, we use a window size of $0.5\,\mathrm{s}$, but the displacement data is input directly this time. An RF is used as a baseline. Both models achieve an OA of over $90\,\%$, with the CNN reaching $94.7\,\%$. We show that the CNN outperforms the RF on a dedicated second dataset when the no-event class consists only of oscillation samples.

Segmentation In the final step, the CNN is evaluated on how well it can segment displacement signals, i.e., detect events from start to end. A threshold-based approach is implemented as a baseline. Overall, the CNN is superior to the baseline regarding the number of detected events and how accurately the event's start and end are captured. Ultimately, we rather overestimate the event duration as otherwise important information might be lost. While the CNN mainly acts accordingly, occasionally, parts of an event are missed. We also show that when overestimating the duration too much it can lead to challenges with the follow-up methods of MONITOR. Another weak point of the CNN segmentation is that if the window falls in the lowest point of an event, it would usually misclassify it as no event. Consequently, the event would be split into two parts. Overall, however, we successfully achieved research goal I and were able to detect vehicle crossings.

8.1.2 Conclusions for Part II

Part II focuses on characterizing extracted events. Naturally, this step succeeds in the event detection in MONITOR. We rely on reference data labeled via UAV video data to develop adequate methods. Upon the collected dataset, events are first classified according to the number of vehicles involved, with the overall objective of identifying single-vehicle crossings. We define a single event as one where, at no point during a vehicle's crossing, another vehicle is on top of the monitored bridge field. Otherwise, we speak of a multi event. Afterward, several vehicle parameters are extracted from single events.

Vehicle Count We conduct two different studies with the same goal of estimating the vehicle count within events. One differentiates between single and multiple events; the other tries to extract the exact amount of vehicles involved in an event. We use data-driven methods in both tasks by applying three models: MiniRocket,

an RF, and an RF combined with PCA. In order to handle the variable-length displacement, data features are manually extracted for the RF approaches. The raw time series data is passed to MiniRocket as it can handle unequally sized data. Regarding the differentation between single and multi events, MiniRocket outperfoms all other models when applying data augmentation. With a BA of $90\,\%$ the model achieves satisfying results. In the second task, estimating the exact number of vehicles, RF without data augmentation had the best results. In fact, interpreting its predictions as a single vs. multi event classification by combining all classes with more than one vehicle also outperforms the MiniRocket of the previous task.
For that reason, we use the RF without data augmentation in MONITOR. Here, we have a trade-off: We try to overestimate the event during event detection but train the single event RF on perfectly extracted data. As a consequence, many single event trucks are falsely classified as multi events in our MONITOR test. Moreover, since we determine vehicle parameters only on single events in this thesis, they are dropped unnecessarily. On the other hand, no multi events are misclassified as single events. This is good, as otherwise subsequent methods would produce non-sensical results.

Data Augmentation UAV flights are short due to the fast battery drainage. Consequently, our dataset is small. Moreover, the dataset is imbalanced in different ways. For instance, cars are more frequent compared to trucks, and events with only one car occur more often than events with several vehicles. Finally, depending on the task, there is a skew in the input data. Multi events generally take longer than single events, e.g., when vehicles drive in a row. This might not be the case when crossings are simultaneous due to opposing driving directions. In order to handle this small, imbalanced, and skewed dataset, we apply data augmentation in tasks related to the vehicle count. We undersample multi events and oversample single events in the temporal dimension. Additional data points are generated by rescaling the displacement of each event.
The effect of our data augmentation on an RF is analyzed using SHAP, an ad-hoc model-agnostic method of Explainable AI. Without augmentation, the RF overwhelmingly relies on the event duration when distinguishing single events from multi events. Trained on augmented data, the group of features the RF considers for predictions is more versatile. The event duration's importance decreases drastically. Thus, our data augmentation is successful. Concerning the classification results, the impact is minimal. The OA improves with augmentation. However, since the dataset is small, critical data points are barely represented in the test set.

Transferability We have monitored two bridges throughout this work. Thus, transferability can be evaluated to a small extent. To this end, we train our single vs. multi event models only on Bridge A and apply it to Bridge B data blindly. For comparison, they are also trained with both datasets combined. The blind transfer works when no data augmentation is used. This makes sense, as the difference in duration between single and multi events is a common trait. With data augmentation, the results drop considerably on Bridge B. Other features are less easily transferable to different bridges than the event duration. Combining the datasets improves results. However, a definite conclusion is challenging to draw with only two bridges and few data points.

Vehicle Parameter Extraction The final task of this thesis is to determine vehicle parameters from single event displacement data of Bridge A. We target the vehicle type, lane, locus, speed, and the presence of a trailer. Moreover, for vehicles without a trailer, the number of axles and the spacing between them is estimated. Again, we resort to data-driven methods for the extraction of these parameters. As cars account for most of the dataset and as there are fewer variations in car configurations compared to trucks, we also train all methods only on truck events. The classification according to both vehicle type and lane produces satisfying results. Vehicle type classification even performs immaculately. The locus can be estimated with an R^2 of 0.84 using MiniRocket and all vehicles. For the speed, MiniRocket achieves an RMSE of $1.03\,\mathrm{m\,s^{-1}}$ on all vehicles. Trailers can be detected with a BA of at least $82.4\,\%$. All in all, these are very good results. When considering all trailerless vehicles, the axle count can be correctly estimated with a BA of $76.7\,\%$. Its BA drops to $69.4\,\%$ for only trucks as their configurations are more complex. This is still acceptable but the dataset is now even smaller. Finally, the axle spacing can be estimated with an MAE of $0.28\,\mathrm{m}$ for all vehicles and $0.36\,\mathrm{m}$ using only trucks. This is impressive, considering the difficulty of the task.
Applying all these methods in MONITOR reveals the same difficulties as for single event detection. Some parameters require the event duration to be estimated correctly due to the perfectly extracted training data. Thus, their performance drops considerably in MONITOR. Other parameters, such as trailer detection, still work surprisingly well. Ultimately, we could show that data-driven methods can extract vehicle information from bridge displacement data, successfully achieving research goal II.

Filtering for Axle Configuration Identifcation Another interesting finding in this thesis is that high-frequency components in the displacement signal correlate with the presence of axles. Oftentimes, when vehicles cross the eastern junction of Bridge A, vibrations can be seen in the frequency range from $45\,\mathrm{Hz}$ to $65\,\mathrm{Hz}$. We could visualize them by means of bandpass filtering and CWT. While we could not find a generic approach to estimate the axle count or the axle spacing from these vibrations, we exemplified how such an approach could work. The axle spacing determined in this example lies close to the ground truth. To our knowledge, such a relationship has never been shown before using real-world displacement data.

8.2 Potential Advances and Outlook

Our results show the feasibility of the main objective. However, advances can be made in several directions. For instance, the robustness of methods should be increased by collecting more data, i.e., monitoring new bridges or covering a more comprehensive range of environmental influences. Apart from that, two promising enhancements are now discussed.

Event Detection While our event detection approach performs satisfactorily, there are several options to improve our event detection. For one, more postprocessing could be performed. For instance, when the low point of an event is misclassified, the predictions could be repeated after shifting the window slightly to the left and right. Another option is to widen the receptive field of the CNN by increasing the window size. The segment for which the prediction is made could still remain $0.5\,\mathrm{s}$. Of course, this window can also be shortened to achieve a higher resolution for the event segmentation.

Pre-Training As mentioned before, the collection of reference data turns out to be cumbersome. Using pre-training can reduce the amount of labeled data that is needed for training. Briefly summarised, pre-training a model means training it on Task A and then transfering it to Task B with the already trained weights. In our case, we have long displacement measurements without UAV reference data. The therein-included events could be used as training samples for an autoencoder. An autoencoder learns to encode input samples to latent features, and then it reconstructs the original input from the latent space. Thus, no labels are needed while the model learns to process event displacement signals. The encoder can then be used for event classification tasks. This allows us to apply DL methods despite our small dataset. A significant challenge, however, poses the variable length of events during the autoencoder training.

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, et al. “Tensorflow: A system for large-scale machine learning”. In: *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*. 2016, pp. 265–283.

[2] Hamed Habibi Aghdam, Elnaz Jahani Heravi, and others. “Guide to convolutional neural networks”. In: *New York, NY: Springer* 10.978-973 (2017). Publisher: Springer, p. 51.

[3] Sunday Ajala, Harikrishnan Muraleedharan Jalajamony, Midhun Nair, Pradeep Marimuthu, and Renny Edwin Fernandez. “Comparing machine learning and deep learning regression frameworks for accurate prediction of dielectrophoretic force”. en. In: *Scientific Reports* 12.1 (July 2022), p. 11971.

[4] M. Arnold, M. Hoyer, and S. Keller. “Convolutional Neural Networks for Detecting Bridge Crossing Events With Ground-Based Interferometric Radar Data”. en. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* V-1-2021 (June 2021), pp. 31–38.

[5] M. Arnold and S. Keller. “Detection and Classification of Bridge Crossing Events With Gound-Based Interferometric Radar Data and Machine Learning Approaches”. en. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* V-1-2020 (Aug. 2020), pp. 109–116.

[6] M. Arnold and S. Keller. “Machine Learning Approaches for Vehicle Counting on Bridges Based on Global Ground-Based Radar Data”. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* X-2-2024 (2024), pp. 1–8.

[7] Matthias Arnold, Mareike Hoyer, and André Dittrich. “ZEBBRA - Eventbasierte Zustandserfassung und -bewertung von Brücken basierend auf Radar-Sensorik in Kombination mit intelligenten Algorithmen : Abschlussbericht : Laufzeit: 01.08.2018-31.12.2021”. In: (2022).

[8] Matthias Arnold and Sina Keller. “Machine Learning and Signal Processing for Bridge Traffic Classification with Radar Displacement Time-Series Data”. en. In: *Infrastructures* 9.3 (Mar. 2024). Number: 3 Publisher: Multidisciplinary Digital Publishing Institute, p. 37.

[9] Matthias Arnold and Sina Keller. “Machine-learning for analyzing bridge displacement using radar data”. In: *Bridge Maintenance, Safety, Management, Digitalization and Sustainability*. Vol. 1. CRC Press, June 2024, pp. 1405–1412.

[10] Anthony Bagnall, Jason Lines, Jon Hills, and Aaron Bostrom. “Time-Series Classification with COTE: The Collective of Transformation-Based Ensembles”. In: *IEEE Transactions on Knowledge and Data Engineering* 27.9 (Sept. 2015). Conference Name: IEEE Transactions on Knowledge and Data Engineering, pp. 2522–2535.

[11] Marília Barandas, Duarte Folgado, Letícia Fernandes, et al. “TSFEL: Time Series Feature Extraction Library”. In: *SoftwareX* 11 (Jan. 2020), p. 100456.

[12] David R. Bassett, Lindsay P. Toth, Samuel R. LaMunion, and Scott E. Crouter. “Step Counting: A Review of Measurement Considerations and Health-Related Applications”. en. In: *Sports Medicine* 47.7 (July 2017), pp. 1303–1315.

[13] Mohamed Amine Belkadi, Abdelhamid Daamouche, and Farid Melgani. “A deep neural network approach to QRS detection using autoencoders”. In: *Expert Systems with Applications* 184 (Dec. 2021), p. 115528.

[14] G. Bernardini, G. De Pasquale, N. Gallino, and C. Gentile. “Microwave interferometer for ambient vibration measurements on civil engineering structures: 2. Application to full-scale Bridges”. In: *Experimental Vibration Analysis of Civil Engineering Structures (EVACES’07)*. 2007, pp. 153–162.

[15] Sizhen Bian, Vitor F Rey, Peter Hevesi, and Paul Lukowicz. “Passive Capacitive based Approach for Full Body Gym Workout Recognition and Counting”. In: *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom*. ISSN: 2474-249X. Mar. 2019, pp. 1–10.

[16] G. Bianchi and R. Sorrentino. *Electronic Filter Simulation & Design*. McGraw-Hill Education, 2007.

[17] Agnieszka Bier, Agnieszka Jastrzebska, and Paweł Olszewski. “Variable-Length Multivariate Time Series Classification Using ROCKET: A Case Study of Incident Detection”. In: *IEEE Access* 10 (2022). Conference Name: IEEE Access, pp. 95701–95715.

[18] Leo Breiman. “Random Forests”. en. In: *Machine Learning* 45.1 (Oct. 2001), pp. 5–32.

[19] Stephen Butterworth and others. “On the theory of filter amplifiers”. In: *Wireless Engineer* 7.6 (1930), pp. 536–541.

[20] Danilo Bzdok, Naomi Altman, and Martin Krzywinski. “Statistics versus machine learning”. In: *Nature methods* 15.4 (2018). Publisher: NIH Public Access, p. 233.

[21] Wenjie Cai and Danqin Hu. “QRS Complex Detection Using Novel Deep Learning Neural Networks”. en. In: *IEEE Access* 8 (2020), pp. 97082–97089.

[22] Giovanni Cerulli. *Fundamentals of Supervised Machine Learning: With Applications in Python, R, and Stata*. Springer Nature, 2023.

[23] Hugh Chen, Joseph D. Janizek, Scott Lundberg, and Su-In Lee. *True to the Model or True to the Data?* en. arXiv:2006.16234 [cs, stat]. June 2020.

[24] Szi-Wen Chen, Hsiao-Chen Chen, and Hsiao-Lung Chan. “A real-time QRS detection method based on moving-averaging incorporating with wavelet denoising”. In: *Computer Methods and Programs in Biomedicine* 82.3 (June 2006), pp. 187–195.

[25] Yanping Chen, Adena Why, Gustavo Batista, Agenor Mafra-Neto, and Eamonn Keogh. “Flying Insect Classification with Inexpensive Sensors”. en. In: *Journal of Insect Behavior* 27.5 (Sept. 2014), pp. 657–677.

[26] Kukjin Choi, Jihun Yi, Changhwa Park, and Sungroh Yoon. “Deep Learning for Anomaly Detection in Time-Series Data: Review, Analysis, and Guidelines”. en. In: *IEEE Access* 9 (2021), pp. 120043–120065.

[27] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W. Kempa-Liehr. “Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package)”. In: *Neurocomputing* 307 (Sept. 2018), pp. 72–77.

[28] T. Cover and P. Hart. “Nearest neighbor pattern classification”. In: *IEEE Transactions on Information Theory* 13.1 (Jan. 1967). Conference Name: IEEE Transactions on Information Theory, pp. 21–27.

[29] Utpal Kumar Das, Kok Soon Tey, Mehdi Seyedmahmoudian, et al. “Forecasting of photovoltaic power generation and model optimization: A review”. In: *Renewable and Sustainable Energy Reviews* 81 (Jan. 2018), pp. 912–928.

[30] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, et al. *The UCR Time Series Archive*. en. arXiv:1810.07758 [cs, stat]. Sept. 2019.

[31] Gaetano De Pasquale, Giulia Bernardini, Pier Paolo Ricci, and Carmelo Gentile. “Ambient vibration testing of bridges by non-contact microwave interferometer”. In: *2008 IEEE Radar Conference*. ISSN: 2375-5318. May 2008, pp. 1–6.

[32] Rina Dechter. “Learning while searching in constraint-satisfaction problems”. In: (1986). Publisher: University of California, Computer Science Department, Cognitive Systems . . .

[33] Devis Dei, Daniele Mecatti, and Massimiliano Pieraccini. “Static Testing of a Bridge Using an Interferometric Radar: The Case Study of “Ponte degli Alpini,” Belluno, Italy”. en. In: *The Scientific World Journal* 2013 (Oct. 2013). Publisher: Hindawi, e504958.

[34] Angus Dempster. “ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels”. en. In: *Data Mining and Knowledge Discovery* 34 (2020), p. 42.

[35] Angus Dempster, Daniel F. Schmidt, and Geoffrey I. Webb. “MiniRocket: A Very Fast (Almost) Deterministic Transform for Time Series Classification”. en. In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. Virtual Event Singapore: ACM, Aug. 2021, pp. 248–257.

[36] Jay L. Devore, Kenneth N. Berk, and Matthew A. Carlton. *Modern mathematical statistics with applications*. Third Edition. Springer texts in statistics. Springer, 2021.

[37] Mahidhar Dwarampudi and N. V. Subba Reddy. *Effects of padding on LSTMs and CNNs*. en. arXiv:1903.07288 [cs, stat]. Mar. 2019.

[38] A. Döring, M. Vogelbacher, O. Schneider, et al. “Ratio-based features for bridge damage detection based on displacement influence line and curvature influence line”. In: *Bridge Safety, Maintenance, Management, Life-Cycle, Resilience and Sustainability*. Num Pages: 9. CRC Press, 2022.

[39] Muhammad Fachrie and others. “A simple vehicle counting system using deep learning with YOLOv3 model”. In: *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)* 4.3 (2020), pp. 462–468.

[40] Mengting Fang, Zhenglong Zhou, Sharon Chen, and Jay McClelland. “Can a recurrent neural network learn to count things?” In: *CogSci*. Vol. 6. 2018, pp. 360–365.

[41] Amir-massoud Farahmand, Sepideh Pourazarm, and Daniel Nikovski. “Random Projection Filter Bank for Time Series Data”. en. In: *Advances in neural information processing systems* 30 (2017).

[42] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. “Deep learning for time series classification: a review”. en. In: *Data Mining and Knowledge Discovery* 33.4 (July 2019), pp. 917–963.

[43] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, et al. “InceptionTime: Finding AlexNet for time series classification”. en. In: *Data Mining and Knowledge Discovery* 34.6 (Nov. 2020), pp. 1936–1962.

[44] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. “Unsupervised Scalable Representation Learning for Multivariate Time Series”. en. In: *Advances in neural information processing systems* 32 (2019).

[45] Yoav Freund and Robert E. Schapire. “A desicion-theoretic generalization of on-line learning and an application to boosting”. en. In: *Computational Learning Theory*. Vol. 904. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 23–37.

[46] Ursula Freundt, Sebastian Böning, and Rolf Kaschner. “Straßenbrücken zwischen aktuellem und zukünftigem Verkehr – Straßenverkehrslasten nach DIN EN 1991-2/NA”. de. In: *Beton- und Stahlbetonbau* 106.11 (2011), pp. 736–746.

[47] Jerome H. Friedman. “Greedy Function Approximation: A Gradient Boosting Machine”. In: *The Annals of Statistics* 29.5 (2001). Publisher: Institute of Mathematical Statistics, pp. 1189–1232.

[48] Kunihiko Fukushima. “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position”. In: *Biological cybernetics* 36.4 (1980). Publisher: Springer, pp. 193–202.

[49] Ben D. Fulcher and Nick S. Jones. “Highly Comparative Feature-Based Time-Series Classification”. In: *IEEE Transactions on Knowledge and Data Engineering* 26.12 (Dec. 2014). Conference Name: IEEE Transactions on Knowledge and Data Engineering, pp. 3026–3037.

[50] Timon Gehr, Sasa Misailovic, and Martin Vechev. "PSI: Exact symbolic inference for probabilistic programs". In: *Computer Aided Verification: 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part I 28*. Springer, 2016, pp. 62–83.

[51] Carmelo Gentile and Giulia Bernardini. "An interferometric radar for non-contact measurement of deflections on civil engineering structures: laboratory and full-scale tests". en. In: *Structure and Infrastructure Engineering* 6.5 (Oct. 2010), pp. 521–534.

[52] Pierre Geurts, Damien Ernst, and Louis Wehenkel. "Extremely randomized trees". en. In: *Machine Learning* 63.1 (Apr. 2006), pp. 3–42.

[53] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep sparse rectifier neural networks". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.

[54] Ahmed Gomaa, Tsubasa Minematsu, Moataz M. Abdelwahab, Mohammed Abo-Zahhad, and Rin-ichiro Taniguchi. "Faster CNN-based vehicle detection and counting strategy for fixed camera scenes". en. In: *Multimedia Tools and Applications* 81.18 (July 2022), pp. 25443–25471.

[55] AD Gordon, L Breiman, JH Friedman, RA Olshen, and Cj J Stone. "Classification and Regression Trees." In: *Biometrics* 40.3 (1984). Publisher: JSTOR, p. 874.

[56] Dörte Haftendorn, Dieter Riebesehl, and Hubert Dammer. *Höhere Mathematik sehen und verstehen*. Springer, 2021.

[57] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". en. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA: IEEE, June 2016, pp. 770–778.

[58] Runnan He, Yang Liu, Kuanquan Wang, et al. "Automatic Detection of QRS Complexes Using Dual Channels Based on U-Net and Bidirectional Long Short-Term Memory". In: *IEEE Journal of Biomedical and Health Informatics* 25.4 (Apr. 2021). Conference Name: IEEE Journal of Biomedical and Health Informatics, pp. 1052–1061.

[59] Wei He, Jifan Liu, Shiqi Song, and Peng Liu. "A non-contact vehicle weighing approach based on bridge weigh-in-motion framework and computer vision techniques". In: *Measurement* 225 (Feb. 2024), p. 113994.

[60] Lars Hertel, Huy Phan, and Alfred Mertins. "Classifying Variable-Length Audio Files with All-Convolutional Networks and Masked Global Pooling". In: *CoRR* abs/1607.02857 (2016).

[61] Donald E Hilt and Donald W Seegrist. *Ridge, a computer program for calculating ridge regression estimates*. Department of Agriculture, Forest Service, Northeastern Forest Experiment, 1977.

[62] K. T. Hsu, C. C. Cheng, and C. H. Chiang. "Long-term monitoring of two highway bridges using microwave interferometer-case studies". In: *2016 16th International Conference on Ground Penetrating Radar (GPR)*. June 2016, pp. 1–5.

[63] G. Hughes. “On the mean accuracy of statistical pattern recognizers”. In: *IEEE Transactions on Information Theory* 14.1 (Jan. 1968). Conference Name: IEEE Transactions on Information Theory, pp. 55–63.

[64] Sio-Song Ieng. “Bridge Influence Line Estimation for Bridge Weigh-in-Motion System”. en. In: *Journal of Computing in Civil Engineering* 29.1 (Jan. 2015), p. 06014006.

[65] Şahin Işık, Bülent Turgut, Yıldıray Anagün, and Murat Olgun. “Predicting Soil Quality Index with a Deep Regression Approach”. In: *Communications in Soil Science and Plant Analysis* 55.9 (May 2024). Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00103624.2024.2305838, pp. 1313–1323.

[66] Kevin Jarrett, Koray Kavukcuoglu, Marc’Aurelio Ranzato, and Yann LeCun. “What is the best multi-stage architecture for object recognition?” In: *2009 IEEE 12th International Conference on Computer Vision*. ISSN: 2380-7504. Sept. 2009, pp. 2146–2153.

[67] Abelino Jiménez and Bhiksha Raj. “Time Signal Classification Using Random Convolutional Features”. In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ISSN: 2379-190X. May 2019, pp. 3592–3596.

[68] Manfred Juretzko. “Reflektorlose Video-Tachymetrie: ein integrales Verfahren zur Erfassung geometrischer und visueller Informationen”. PhD Thesis. Universität Bochum, 2004.

[69] Gerald Kaiser. *A Friendly Guide to Wavelets*. en. Boston: Birkhäuser, 2011.

[70] Hamed Kalhori, Mehrisadat Makki Alamdari, Xinqun Zhu, and Bijan Samali. “Nothing-on-Road Axle Detection Strategies in Bridge-Weigh-in-Motion for a Cable-Stayed Bridge: Case Study”. EN. In: *Journal of Bridge Engineering* 23.8 (Aug. 2018). Publisher: American Society of Civil Engineers, p. 05018006.

[71] Fazle Karim, Somshubra Majumdar, and Houshang Darabi. “Insights Into LSTM Fully Convolutional Networks for Time Series Classification”. en. In: *IEEE Access* 7 (2019), pp. 67718–67725.

[72] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Samuel Harford. “Multivariate LSTM-FCNs for Time Series Classification”. en. In: *Neural Networks* 116 (Aug. 2019). arXiv:1801.04503 [cs, stat], pp. 237–245.

[73] T Kawakatsu, K Aihara, A Takasu, and J Adachi. “Deep Sensing Approach to Single-Sensor Bridge Weighing in Motion”. en. In: *Proceedings of the 9th European Workshop on Structural Health Monitoring* (2018).

[74] Takaya Kawakatsu, Kenro Aihara, Atsuhiro Takasu, and Jun Adachi. “Deep Sensing Approach to Single-Sensor Vehicle Weighing System on Bridges”. en. In: *IEEE Sensors Journal* 19.1 (Jan. 2019), pp. 243–256.

[75] Takaya Kawakatsu, Kenro Aihara, Atsuhiro Takasu, and Jun Adachi. “Fully-Neural Approach to Heavy Vehicle Detection on Bridges Using a Single Strain Sensor”. In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ISSN: 2379-190X. May 2020, pp. 3047–3051.

[76] Takaya Kawakatsu, Kenro Aihara, Atsuhiro Takasu, Tomonori Nagayama, and Jun Adachi. “Data-Driven Bridge Weigh-In-Motion”. In: *IEEE Sensors Journal* (2023). Conference Name: IEEE Sensors Journal, pp. 1–1.

[77] Shehroz S. Khan and Ali Abedi. “Step Counting with Attention-based LSTM”. In: *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*. Dec. 2022, pp. 559–566.

[78] Jonathan Kirby. *Spectral methods for the estimation of the effective elastic thickness of the lithosphere*. Springer, 2022.

[79] T. Kohonen. “The self-organizing map”. en. In: *Proceedings of the IEEE* 78.9 (Sept. 1990), pp. 1464–1480.

[80] Max Kuhn and Kjell Johnson. *Applied Predictive Modeling*. SpringerLink. Springer New York, NY, 2013.

[81] Andrew Lansdell, Wei Song, and Brandon Dixon. “Development and testing of a bridge weigh-in-motion method considering nonconstant vehicle speed”. In: *Engineering Structures* 152 (Dec. 2017), pp. 709–726.

[82] B. Lechner, M. Lieschnegg, O. Mariani, M. Pircher, and A. Fuchs. “A Wavelet-Based Bridge Weigh-In-Motion System”. en. In: *International Journal on Smart Sensing and Intelligent Systems* 3.4 (Jan. 2010), pp. 573–591.

[83] Gregory R. Lee, Ralf Gommers, Filip Waselewski, Kai Wohlfahrt, and Aaron O’Leary. “PyWavelets: A Python package for wavelet analysis”. en. In: *Journal of Open Source Software* 4.36 (Apr. 2019), p. 1237.

[84] Timothy P. Lillicrap, Adam Santoro, Luke Marris, Colin J. Akerman, and Geoffrey Hinton. “Backpropagation and the brain”. en. In: *Nature Reviews Neuroscience* 21.6 (June 2020), pp. 335–346.

[85] Jason Lines, Sarah Taylor, and Anthony Bagnall. “Time Series Classification with HIVE-COTE: The Hierarchical Vote Collective of Transformation-Based Ensembles”. In: *ACM Transactions on Knowledge Discovery from Data* 12.5 (July 2018), 52:1–52:35.

[86] Shangqing Liu, Yanchao Zhao, and Bing Chen. “WiCount: A Deep Learning Approach for Crowd Counting Using WiFi Signals”. In: *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*. Dec. 2017, pp. 967–974.

[87] Gilles Louppe. “Understanding random forests”. In: *Cornell University Library* 10 (2014).

[88] Scott M. Lundberg, Gabriel Erion, Hugh Chen, et al. “From local explanations to global understanding with explainable AI for trees”. en. In: *Nature Machine Intelligence* 2.1 (Jan. 2020). Publisher: Nature Publishing Group, pp. 56–67.

[89] Scott M Lundberg and Su-In Lee. “A Unified Approach to Interpreting Model Predictions”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, et al. Vol. 30. Curran Associates, Inc., 2017.

[90] Markus Löning, Anthony Bagnall, Sajaysurya Ganesh, and Viktor Kazakov. “sktime: A Unified Interface for Machine Learning with Time Series”. en. In: (2019).

[91] Rujin Ma, Zhen Zhang, Yiqing Dong, and Yue Pan. “Deep Learning Based Vehicle Detection and Classification Methodology Using Strain Sensors under Bridge Deck”. en. In: *Sensors* 20.18 (Jan. 2020). Number: 18 Publisher: Multidisciplinary Digital Publishing Institute, p. 5051.

[92] F. Magalhães, A. Cunha, and E. Caetano. “Vibration based structural health monitoring of an arch bridge: From automated OMA to damage detection”. en. In: *Mechanical Systems and Signal Processing* 28 (Apr. 2012), pp. 212–228.

[93] Jean Mahowald, Stefan Maas, Viet Ha Nguyen, Danièle Waldmann, and Arno Zuerbes. “Some conclusions from the measurements of temperatures and their gradients on eigenfrequencies of bridges”. In: (2014).

[94] Pankaj Malhotra, Vishnu TV, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. *TimeNet: Pre-trained deep recurrent neural network for time series classification*. en. arXiv:1706.08838 [cs]. June 2017.

[95] Pedro Matias, Duarte Folgado, Hugo Gamboa, and André Carreiro. “Time Series Segmentation Using Neural Networks with Cross-Domain Transfer Learning”. en. In: *Electronics* 10.15 (Jan. 2021). Number: 15 Publisher: Multidisciplinary Digital Publishing Institute, p. 1805.

[96] Warren S McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5 (1943). Publisher: Springer, pp. 115–133.

[97] Gerhard Mehlhorn and Manfred Curbach, eds. *Handbuch Brücken*. de. Wiesbaden: Springer Fachmedien Wiesbaden, 2014.

[98] Martin Meyer. “Signalverarbeitung: analoge und digitale Signale”. In: *Systeme und Filter (German Edition), Vieweg+ Teubner Verlag* (2011).

[99] Lapo Miccinesi, Alessandra Beni, and Massimiliano Pieraccini. “Multi-Monostatic Interferometric Radar for Bridge Monitoring”. en. In: *Electronics* 10.3 (Jan. 2021). Number: 3 Publisher: Multidisciplinary Digital Publishing Institute, p. 247.

[100] Chris Michel and Sina Keller. “Advancing Ground-Based Radar Processing for Bridge Infrastructure Monitoring”. en. In: *Sensors* 21.6 (Jan. 2021). Number: 6 Publisher: Multidisciplinary Digital Publishing Institute, p. 2172.

[101] Chris Michel and Sina Keller. “Determining and Investigating the Variability of Bridges’ Natural Frequencies with Ground-Based Radar”. en. In: (2022), p. 17.

[102] Chris Eric Michel. “Generic Radar Processing Methods for Monitoring Tasks on Bridge Infrastructure”. PhD Thesis. Karlsruher Institut für Technologie (KIT), 2023.

[103] Matthew Middlehurst. “HIVE-COTE 2.0: a new meta ensemble for time series classification”. en. In: *Machine Learning* (2021), p. 33.

[104] Pietro Milillo, Giorgia Giardina, Daniele Perissin, et al. “Pre-Collapse Space Geodetic Observations of Critical Infrastructure: The Morandi Bridge, Genoa, Italy”. en. In: *Remote Sensing* 11.12 (Jan. 2019). Number: 12 Publisher: Multidisciplinary Digital Publishing Institute, p. 1403.

[105] Amin Moghadam, Mohammad AlHamaydeh, and Rodrigo Sarlo. “Dual-purpose procedure for bridge health monitoring and weigh-in-motion used for multiple-vehicle events”. en. In: *Automation in Construction* 148 (Apr. 2023), p. 104768.

[106] Yahya M. Mohammed and Nasim Uddin. “Acceleration-based bridge weigh-in-motion”. en. In: *Bridge Structures* 14.4 (June 2019), pp. 131–138.

[107] Christoph Molnar and Timo Freiesleben. *Supervised Machine Learning For Science: How to stop worrying and love your black box*. 2024.

[108] Douglas C Montgomery, Cheryl L Jennings, and Murat Kulahci. *Introduction to time series analysis and forecasting*. John Wiley & Sons, 2015.

[109] Fred Moses. “Weigh-in-motion system using instrumented bridges”. In: *Transportation Engineering Journal of ASCE* 105.3 (1979). Publisher: American Society of Civil Engineers, pp. 233–249.

[110] Viktor Moskalenko, Nikolai Zolotykh, and Grigory Osipov. “Deep Learning for ECG Segmentation”. en. In: *Advances in Neural Computation, Machine Learning, and Cognitive Research III*. Ed. by Boris Kryzhanovsky, Witali Dunin-Barkowski, Vladimir Redko, and Yury Tiumentsev. Studies in Computational Intelligence. Cham: Springer International Publishing, 2020, pp. 246–254.

[111] Mohsin Munir, Shoaib Ahmed Siddiqui, Andreas Dengel, and Sheraz Ahmed. “DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series”. en. In: *IEEE Access* 7 (2019), pp. 1991–2005.

[112] Linh Nguyen, Dung K. Nguyen, Thang Nguyen, and Truong X. Nghiem. “Convolutional neural network regression for low-cost microalgal density estimation”. In: *e-Prime - Advances in Electrical Engineering, Electronics and Energy* 9 (Sept. 2024), p. 100653.

[113] Skyler Norgaard, Ramyar Saeedi, and Assefaw H. Gebremedhin. “Multi-Sensor Time-Series Classification for Activity Tracking Under Variable Length”. en. In: *IEEE Sensors Journal* 20.5 (Mar. 2020), pp. 2701–2709.

[114] E. J. OBrien, M. J. Quilligan, and R. Karoumi. “Calculating an influence line from direct measurements”. en. In: *Proceedings of the Institution of Civil Engineers - Bridge Engineering* 159.1 (Mar. 2006), pp. 31–34.

[115] T. Ojio, C. H. Carey, E. J. OBrien, C. Doherty, and S. E. Taylor. “Contactless Bridge Weigh-in-Motion”. en. In: *Journal of Bridge Engineering* 21.7 (July 2016), p. 04016032.

[116] G Ososkov and A Shitov. “Gaussian wavelet features and their applications for analysis of discretized signals”. In: *Computer physics communications* 126.1-2 (2000). Publisher: Elsevier, pp. 149–157.

[117] Jiapu Pan and Willis J. Tompkins. “A Real-Time QRS Detection Algorithm”. In: *IEEE Transactions on Biomedical Engineering* BME-32.3 (Mar. 1985). Conference Name: IEEE Transactions on Biomedical Engineering, pp. 230–236.

[118] Debojyoti Paul and Koushik Roy. “Application of bridge weigh-in-motion system in bridge health monitoring: a state-of-the-art review”. en. In: *Structural Health Monitoring* (Mar. 2023), p. 147592172311544.

[119] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, et al. “Scikit-learn: Machine learning in Python”. In: *the Journal of machine Learning research* 12 (2011). Publisher: JMLR. org, pp. 2825–2830.

[120] M. Pieraccini, M. Fratini, F. Parrini, and C. Atzeni. “Dynamic Monitoring of Bridges Using a High-Speed Coherent Radar”. In: *IEEE Transactions on Geoscience and Remote Sensing* 44.11 (Nov. 2006). Conference Name: IEEE Transactions on Geoscience and Remote Sensing, pp. 3284–3288.

[121] Massimiliano Pieraccini, Lapo Miccinesi, Ali Abdorazzagh Nejad, and Azadeh Naderi Nejad Fard. “Experimental Dynamic Impact Factor Assessment of Railway Bridges through a Radar Interferometer”. en. In: *Remote Sensing* 11.19 (Jan. 2019). Number: 19 Publisher: Multidisciplinary Digital Publishing Institute, p. 2207.

[122] John Platt and others. “Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods”. In: *Advances in large margin classifiers* 10.3 (1999). Publisher: Cambridge, MA, pp. 61–74.

[123] Fernando Puente León and Sebastian Bauer. *Praxis der Digitalen Signalverarbeitung. 2. überarb*. KIT Scientific Publishing, 2017.

[124] Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Numerical Mathematics*. Vol. 37. Jan. 2007.

[125] Oona Rainio, Jarmo Teuho, and Riku Klén. “Evaluation metrics and statistical tests for machine learning”. en. In: *Scientific Reports* 14.1 (Mar. 2024). Publisher: Nature Publishing Group, p. 6086.

[126] Felix M. Riese. “Development and Applications of Machine Learning Methods for Hyperspectral Data”. PhD Thesis. Karlsruher Institut für Technologie (KIT), 2020.

[127] Felix M. Riese. *SuSi: SUpervised Self-organIzing maps in Python*. `https://doi.org/10.5281/zenodo.2609130doi.org/10.5281/zenodo.2609130`. 2019.

[128] Felix M. Riese, Sina Keller, and Stefan Hinz. “Supervised and Semi-Supervised Self-Organizing Maps for Regression and Classification Focusing on Hyperspectral Data”. en. In: *Remote Sensing* 12.1 (Jan. 2020). Number: 1 Publisher: Multidisciplinary Digital Publishing Institute, p. 7.

[129] Angela Lopez-del Rio, Maria Martin, Alexandre Perera-Lluna, and Rabie Saidi. “Effect of sequence padding on the performance of deep learning models in archaeal protein functional prediction”. en. In: *Scientific Reports* 10.1 (Sept. 2020), p. 14634.

[130] Christian Robert. *Machine learning, a probabilistic perspective*. 2014.

[131] Ribana Roscher, Bastian Bohn, Marco F. Duarte, and Jochen Garcke. “Explainable Machine Learning for Scientific Insights and Discoveries”. In: *IEEE Access* 8 (2020). Conference Name: IEEE Access, pp. 42200–42216.

[132] Frank Rosenblatt and others. *Principles of neurodynamics: Perceptrons and the theory of brain mechanisms*. Vol. 55. Spartan books Washington, DC, 1962.

[133] Alejandro Pasos Ruiz, Michael Flynn, James Large, Matthew Middlehurst, and Anthony Bagnall. “The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances”. en. In: *Data Mining and Knowledge Discovery* 35.2 (Mar. 2021), pp. 401–449.

[134] AK Md Ehsanes Saleh, Mohammad Arashi, and BM Golam Kibria. *Theory of ridge regression estimation with applications*. John Wiley & Sons, 2019.

[135] Harald Schmid. *Mathematik für Ingenieurwissenschaften: Vertiefung*. Springer, 2022.

[136] Manuel Schneider, Norbert Greifzu, Lei Wang, et al. “An end-to-end machine learning approach with explanation for time series with varying lengths”. en. In: *Neural Computing and Applications* (Feb. 2024).

[137] Patrick Schäfer. “Scalable time series classification”. en. In: *Data Mining and Knowledge Discovery* 30.5 (Sept. 2016), pp. 1273–1298.

[138] Patrick Schäfer. “The BOSS is concerned with time series classification in the presence of noise”. en. In: *Data Mining and Knowledge Discovery* 29.6 (Nov. 2015), pp. 1505–1530.

[139] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. “Financial time series forecasting with deep learning : A systematic literature review: 2005–2019”. In: *Applied Soft Computing* 90 (May 2020), p. 106181.

[140] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. en. 1st ed. Cambridge University Press, May 2014.

[141] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. “Activation functions in neural networks”. In: *Towards Data Sci* 6.12 (2017), pp. 310–316.

[142] Pedro Silva, Eduardo Luz, Elizabeth Wanner, David Menotti, and Gladston Moreira. “QRS Detection in ECG Signal with Convolutional Network”. en. In: *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. Ed. by Ruben Vera-Rodriguez, Julian Fierrez, and Aythami Morales. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pp. 802–809.

[143] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. en. arXiv:1409.1556 [cs]. Apr. 2015.

[144] Georgios Sopidis, Michael Haslgrübler, and Alois Ferscha. “Counting Activities Using Weakly Labeled Raw Acceleration Data: A Variable-Length Sequence Approach with Deep Learning to Maintain Event Duration Flexibility”. en. In: *Sensors* 23.11 (Jan. 2023). Number: 11 Publisher: Multidisciplinary Digital Publishing Institute, p. 5057.

[145] Saber Taghvaeeyan and Rajesh Rajamani. “Portable Roadside Sensors for Vehicle Counting, Classification, and Speed Measurement”. In: *IEEE Transactions on Intelligent Transportation Systems* 15.1 (Feb. 2014). Conference Name: IEEE Transactions on Intelligent Transportation Systems, pp. 73–83.

[146] Chang Wei Tan, Angus Dempster, Christoph Bergmeir, and Geoffrey I. Webb. “MultiRocket: multiple pooling operators and transformations for fast and effective time series classification”. en. In: *Data Mining and Knowledge Discovery* 36.5 (Sept. 2022), pp. 1623–1646.

[147] Vignesh Thakkar, Suman Tewary, and Chandan Chakraborty. “Batch Normalization in Convolutional Neural Networks — A comparative study with CIFAR-10 data”. en. In: *2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT)*. Kolkata: IEEE, Jan. 2018, pp. 1–5.

[148] Lin Tzuta. *LabelImg*. https://github.com/tzutalin/labelImg. 2015.

[149] Saidrasul Usmankhujaev, Bunyodbek Ibrokhimov, Shokhrukh Baydadaev, and Jangwoo Kwon. “Time Series Classification with InceptionFCN”. en. In: *Sensors* 22.1 (Jan. 2022). Number: 1 Publisher: Multidisciplinary Digital Publishing Institute, p. 157.

[150] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[151] Digna R. Velez, Bill C. White, Alison A. Motsinger, et al. “A balanced accuracy function for epistasis modeling in imbalanced datasets using multifactor dimensionality reduction”. en. In: *Genetic Epidemiology* 31.4 (2007), pp. 306–315.

[152] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272.

[153] Zhiguang Wang, Weizhong Yan, and Tim Oates. “Time series classification from scratch with deep neural networks: A strong baseline”. In: *2017 International Joint Conference on Neural Networks (IJCNN)*. ISSN: 2161-4407. May 2017, pp. 1578–1585.

[154] Jacek Welc and Pedro J Rodriguez Esquerdo. “Applied regression analysis for business”. In: *Cham: Springer* (2018). Publisher: Springer.

[155] Hui Xiong, Meiling Liang, and Jinzhen Liu. “A Real-Time QRS Detection Algorithm Based on Energy Segmentation for Exercise Electrocardiogram”. en. In: *Circuits, Systems, and Signal Processing* 40.10 (Oct. 2021), pp. 4969–4985.

[156] Yang Yu, CS Cai, and Lu Deng. “State-of-the-art review on bridge weigh-in-motion technology”. en. In: *Advances in Structural Engineering* 19.9 (Sept. 2016). Publisher: SAGE Publications Ltd STM, pp. 1514–1530.

[157] Yang Yu, Cs Cai, and Lu Deng. “Vehicle axle identification using wavelet analysis of bridge global responses”. en. In: *Journal of Vibration and Control* 23.17 (Oct. 2017), pp. 2830–2840.

[158] Kivilcim Yuksel, Damien Kinet, Karima Chah, and Christophe Caucheteur. “Implementation of a Mobile Platform Based on Fiber Bragg Grating Sensors for Automotive Traffic Monitoring”. en. In: *Sensors* 20.6 (Mar. 2020), p. 1567.

[159] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. “A Transformer-based Framework for Multivariate Time Series Representation Learning”. en. In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. Virtual Event Singapore: ACM, Aug. 2021, pp. 2114–2124.

[160] Xuchao Zhang, Yifeng Gao, Jessica Lin, and Chang-Tien Lu. “TapNet: Multivariate Time Series Classification with Attentional Prototypical Network”. en. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.04 (Apr. 2020). Number: 04, pp. 6845–6852.

[161] Hua Zhao, Chengjun Tan, Eugene J. OBrien, Nasim Uddin, and Bin Zhang. “Wavelet-Based Optimum Identification of Vehicle Axles Using Bridge Measurements”. en. In: *Applied Sciences* 10.21 (Jan. 2020). Number: 21 Publisher: Multidisciplinary Digital Publishing Institute, p. 7485.

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

List of Figures

List of Tables

List of Abbreviations

AB	AdaBoost
AI	Artificial Intelligence
ANN	Artificial Neural Network
BA	Balanced Accuracy
BOSS	Bag-of-SFA-Symbols
BWIM	Bridge Weigh-in-Motion
CAM	Class Activation Map
CNN	Convolutional Neural Network
CWT	Continuous Wavelet Transform
COTE	Collective of Transformation-Based Ensembles
DL	Deep Learning
DTW	Dynamic Time Warping
ECG	Electrocardiogram
ET	Extremely Randomized Trees
FAD	free-of-axle-detector
FC	Fully-Connected
FCN	Fully-Convolutional Network
FEM	Finite element method
FN	False negative
FP	False positive
GB	Gradient Boosting
GBR	Ground-Based Radar

GUI	Graphical User Interface
HIVE-COTE 2.0	Hierarchical Vote Collective of Transformation-based Ensembles 2.0
IIR	Infinite Impulse Response
IBIS	Image by Interferometric Survey
HIVE-COTE	Hierarchical Vote Collective of Transformation-based Ensembles
KDE	Kernel Density Estimation
KIT-IPF	Institut für Photogrammetrie und Fernerkundung
KNN	k-Nearest-Neighbours
LOS	line of sight
LP	low-pass
LR	Linear Regression
LSTM	Long Short Term Memory
MAD	Mean Absolute Deviation
MAE	Mean Average Error
MIMO	multiple-input and multiple-output
MiniRocket	Minimally Random Convolutional Kernel Transform
ML	Machine Learning
MONITOR	Methods for data-driven detection and classification of vehicles in non-invasive radar data
MSDMV	Mean Squared Deviation from the Mean Value
MSE	Mean Squared Error
NLP	Natural Language Processing
NOR	Nothing-on-Road
OA	Overall Accuracy
P	Precision
PCA	Principal Component Analysis

p.p.	percentage points
PPV	Proportion of Positive Values
R^2	Coefficient of Determination
RC	Recall
ReLU	Rectified Linear Unit
ResNet	Residual Network
RMSDMV	Root Mean Squared Deviation from the Mean Value
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
RF	Random Forest
ROCKET	Random Convolutional Kernel Transform
SFA	Symbolic Fourier Approximation
SHAP	Shapley Additive Explanations
SHM	Structural Health Montoring
SNR	signal-to-noise ratio
SOM	Self-Organizing Map
SVM	Support Vector Machine
UAV	Unmanned Aerial Vehicle
TN	True negative
TP	True positive

Supplementary Material

A

A.1 Bridge A Example Events

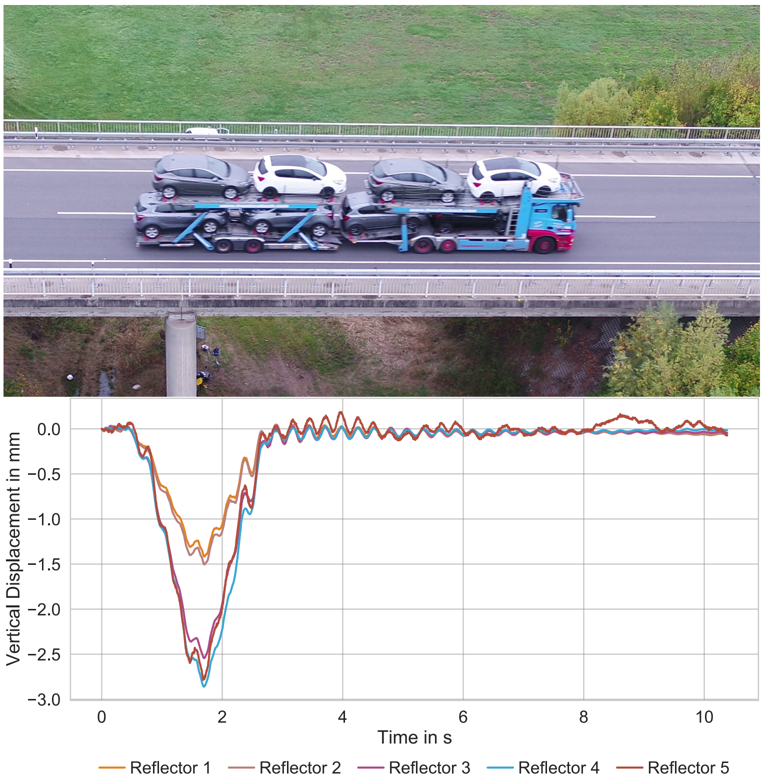


Fig. A.1.: Bridge A example truck on lane 1.

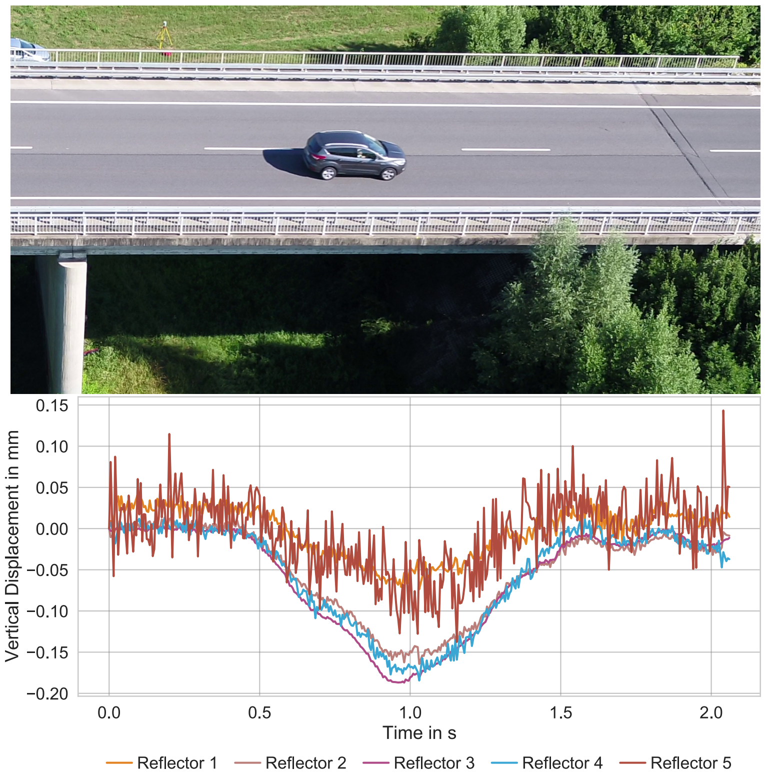


Fig. A.2.: Bridge A example car on lane 1.

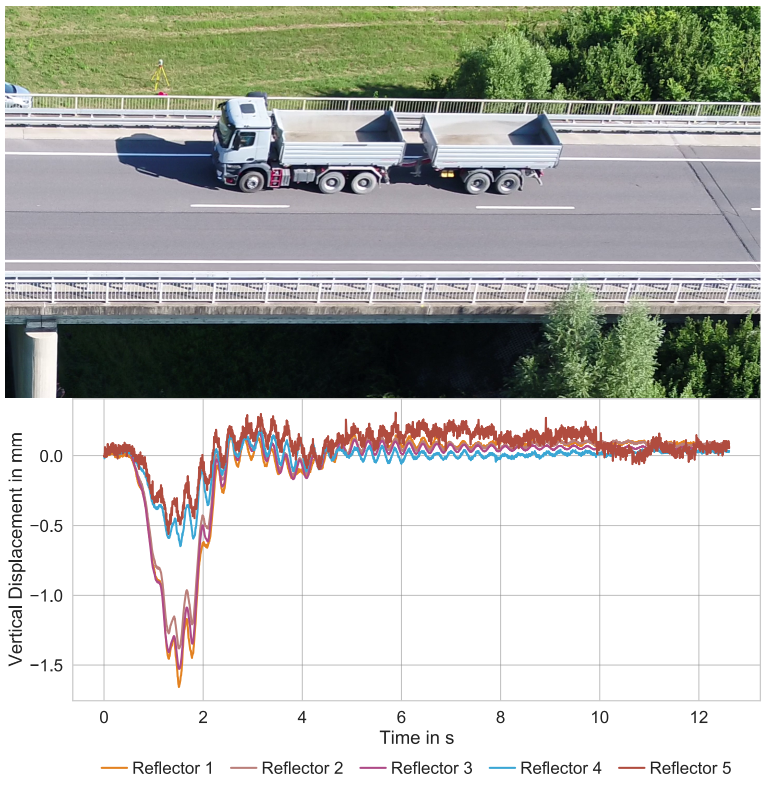


Fig. A.3.: Bridge A example truck on lane 2.

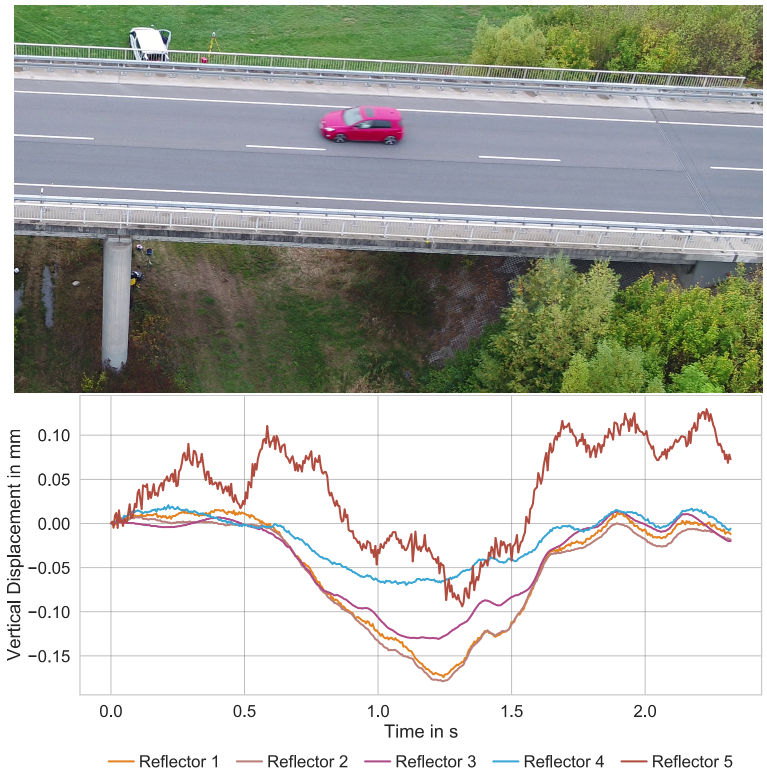


Fig. A.4.: Bridge A example car on lane 2.

A.2 Bridge B Example Events

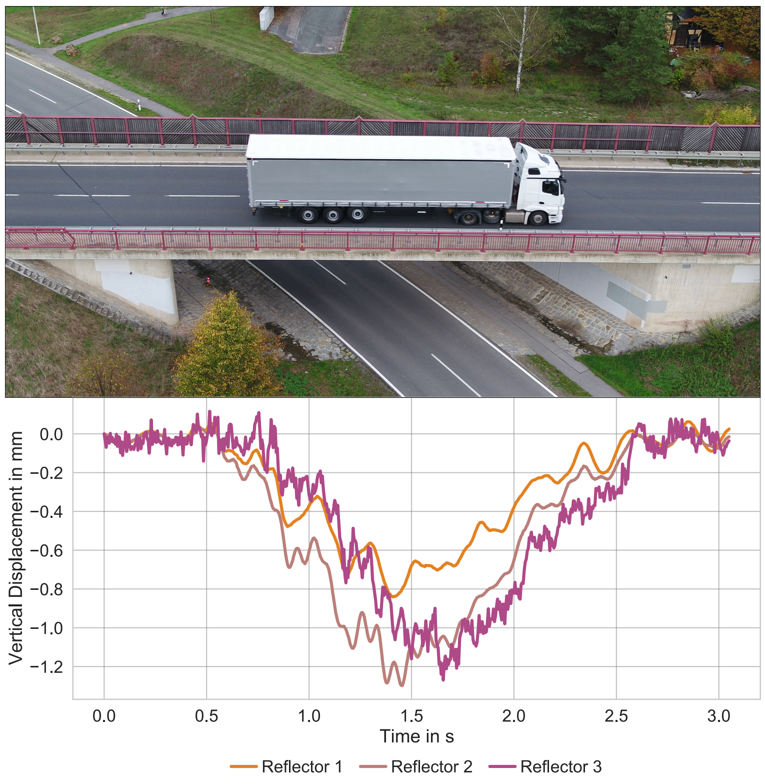


Fig. A.5.: Bridge B example truck on lane 1.

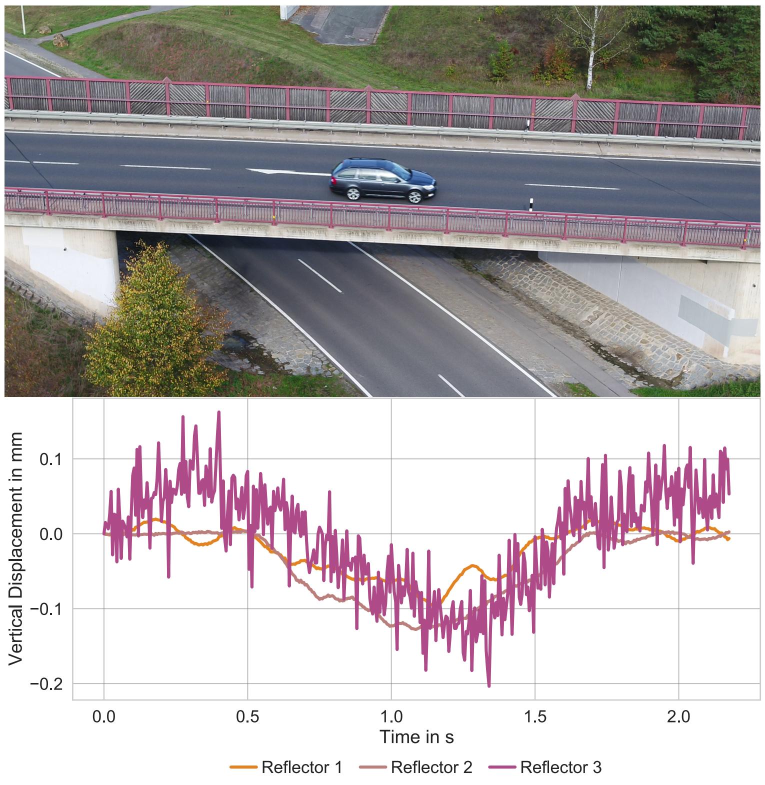


Fig. A.6.: Bridge B example car on lane 1.

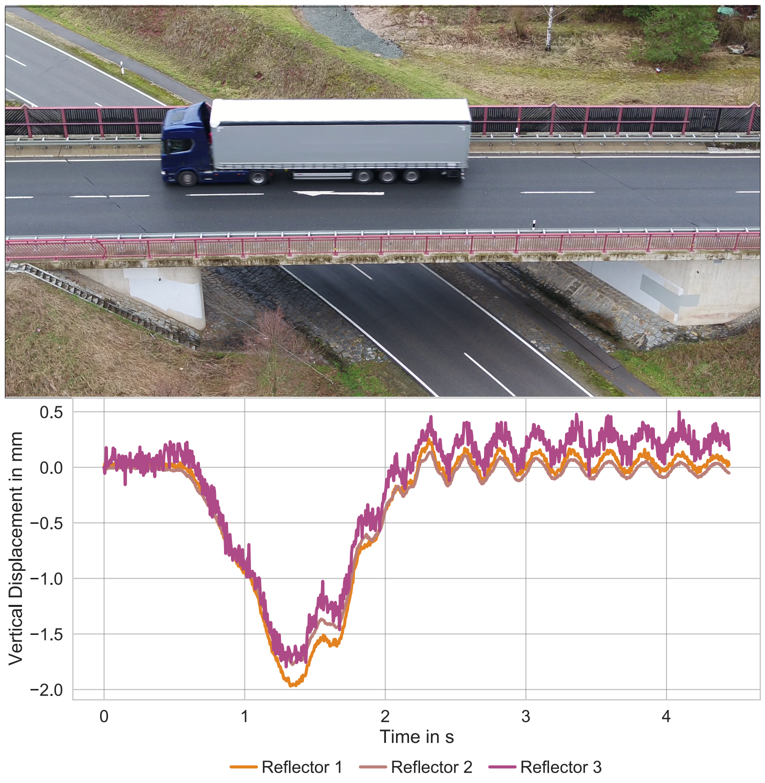


Fig. A.7.: Bridge B example truck on lane 2.

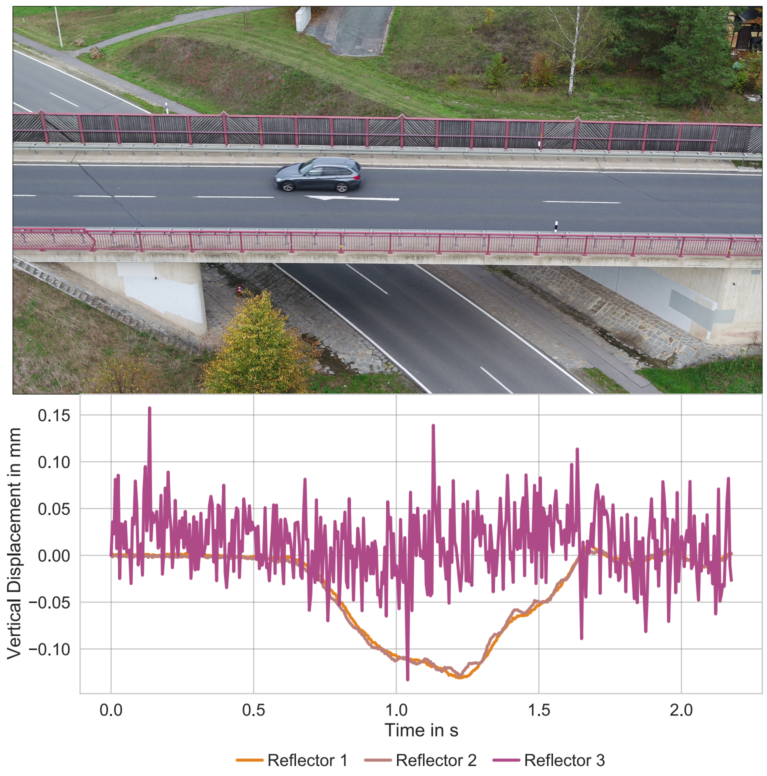


Fig. A.8.: Bridge B example car on lane 2.

A.3 Examples for Filter-Based Axle Configuration Identification

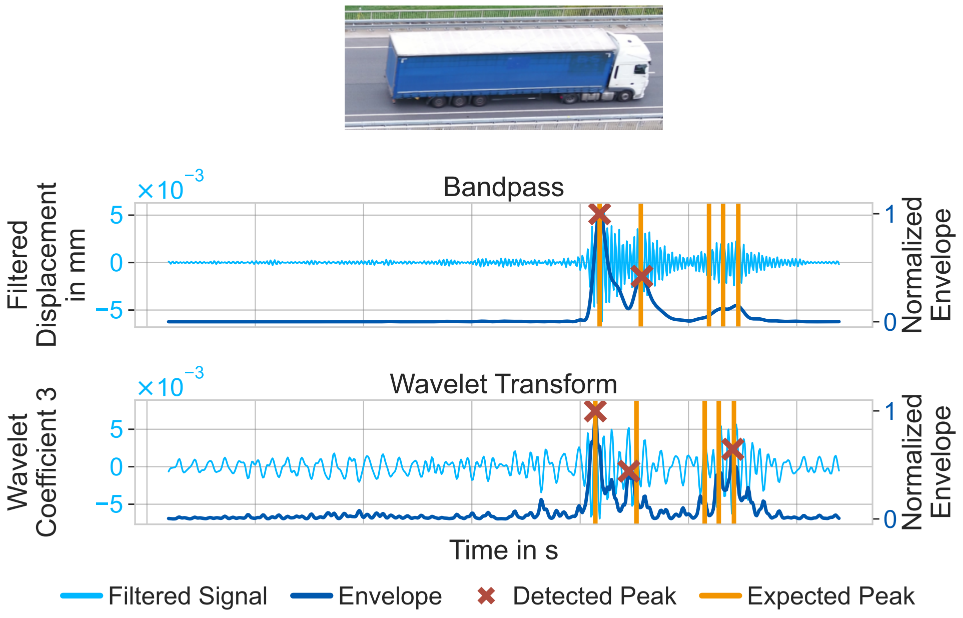


Fig. A.9.: The truck that we used in Figure 2.3 as an example for a Bridge A event.

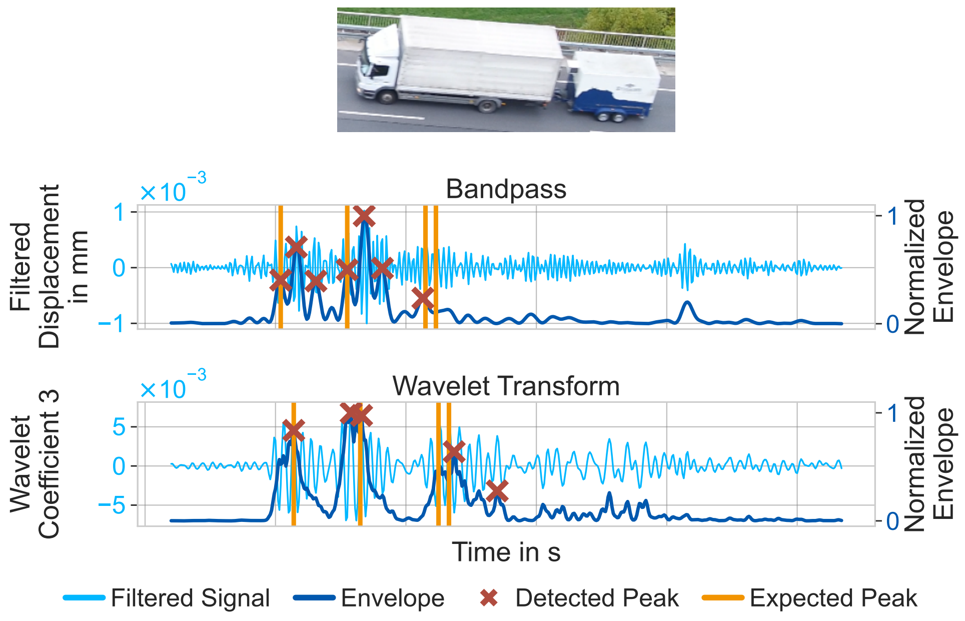


Fig. A.10.: A truck with a unusal small trailer.

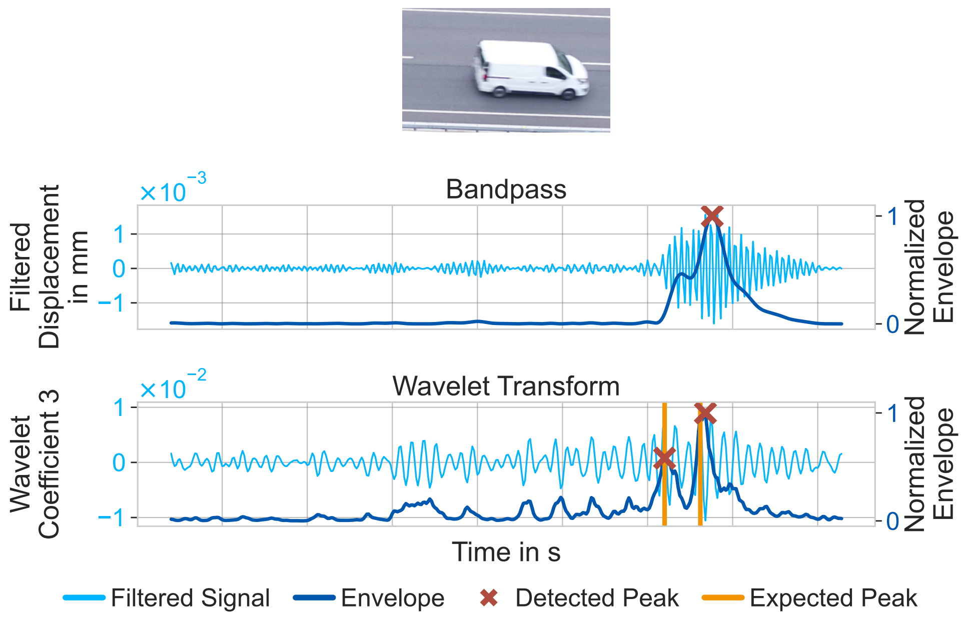


Fig. A.11.: Example of a car in which axles are recognizable.

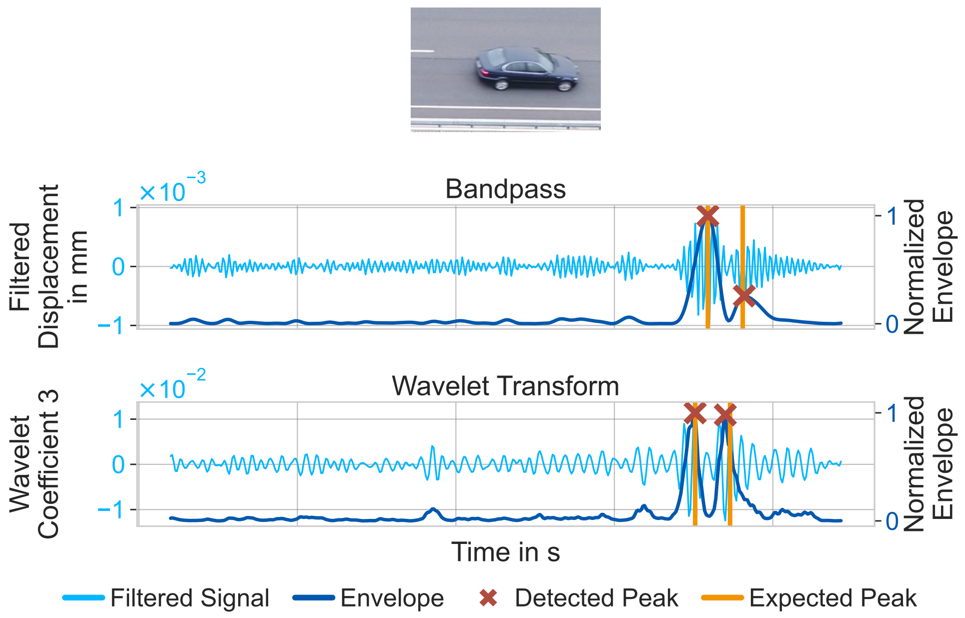


Fig. A.12.: Another example of a car in which axles are recognizable.

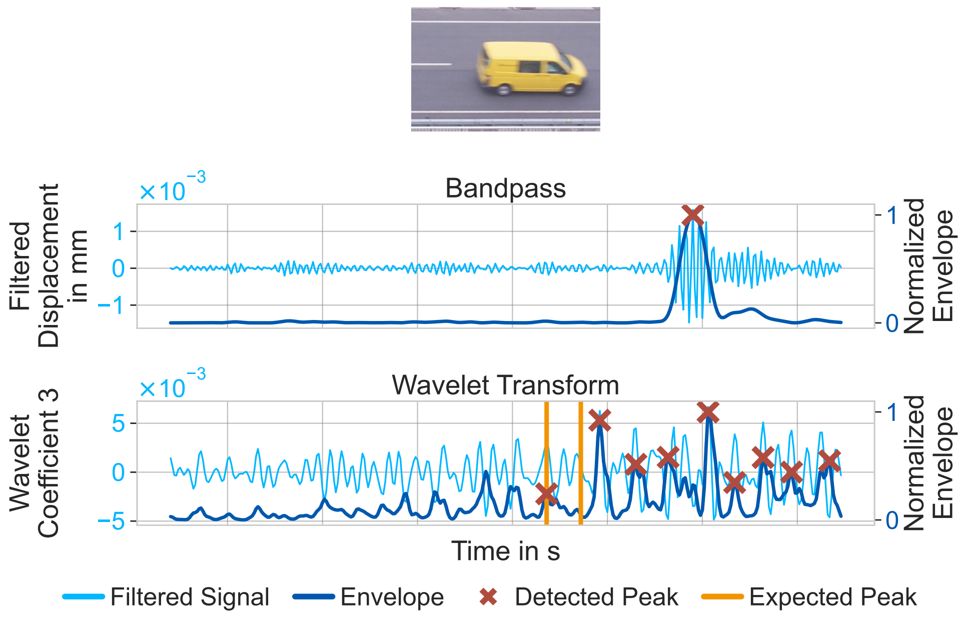


Fig. A.13.: Example of a car in which the CWT does not result in usable peaks.

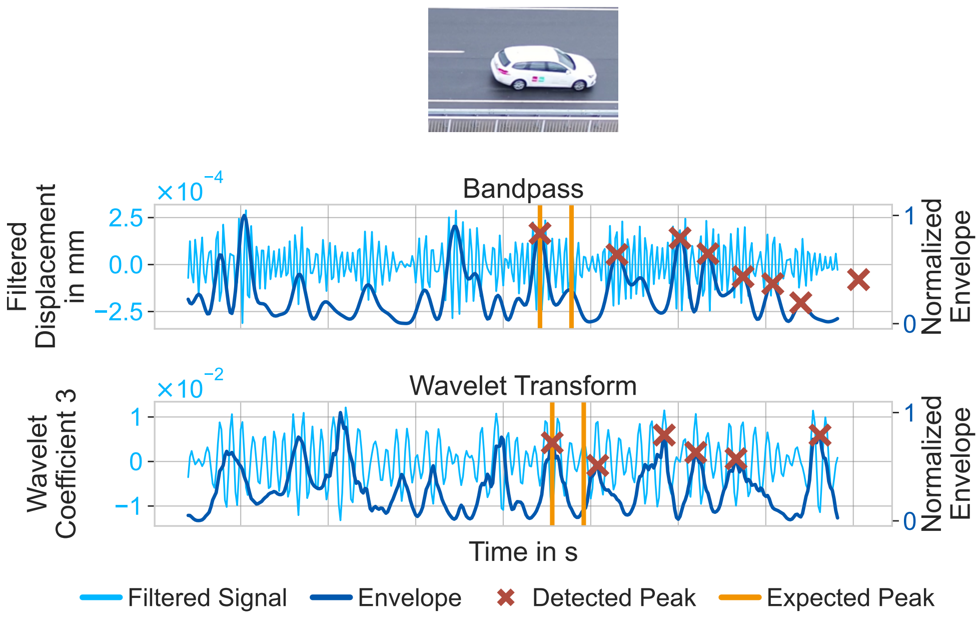


Fig. A.14.: Example of a car for which neither method leads to meaningful peaks.

Colophon

This thesis was typeset with $\LaTeX\ 2_\varepsilon$. It uses the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at `http://cleanthesis.der-ric.de/`.