UNIVERSITÄT SIEGEN

Anomaly Detection and Event Recognition in Cars based on Multimodal Sensor Data Interpretation

DISSERTATION
to obtain the degree of
Doctor of Engineering

submitted by Hawzhin Hozhabr Pour

submitted to the School of Science and Technology
of the University of Siegen
Siegen 2023

Supervisor and first appraiser
Prof. Dr. Roland Wismüller
University of Siegen

Second appraiser
Prof. Dr. Marcin Grzegorzek
University of Lübeck

Date of the oral examination: 06.03.2024

Acknowledgements

I want to express my gratitude to my direct Ph.D. supervisor, Prof. Dr. Roland Wismüller. His warm acceptance into his research group, abundant opportunities to delve into fascinating projects in pattern recognition, and unwavering professional and personal support have been the cornerstone of my journey.

A special thank you to my second supervisor, Prof. Dr. Marcin Grzegorzek. Your guidance and advice have been a beacon throughout my research and career. Beyond being a professional mentor, I've been inspired by your positive outlook on the world, providing a ray of light in challenging situations. With your incredible guidance, this work was a possible feat.

I also sincerely appreciate Dr. Sebastian Fudickar and the entire junior research group (Move-Group) at the University of Lübeck for their patient supervision and collaboration. A special thanks to Dr. Frederic Li for his endless support. To my colleagues at the Research Group of Operating Systems and Distributed Systems at the University of Siegen, especially Regina Syska, your support has felt like a caring embrace, much like that of a devoted mother, throughout my PhD journey at Siegen.

None of this would have been possible without the love and support of my parents. To Hana and Hemen, my sister and brother, your unconditional love is the foundation of everything I have achieved. Thank you to my dearest friend, Dr. Rojair Penjweini, for sharing every moment of this lengthy journey and providing unwavering support. Leszek Podwojci, your support during this Ph.D. journey has been like that of a family and finally, to Julika and Annette Nakajima for being there whenever I needed assistance.

This work was predominantly funded by two key projects: LEICAR, supported by the German Federal Ministry of Education and Research (BMBF) under grant number 01IS15048B, and GEMIMEG-II, funded by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) under grant number 01MT20001L.

Hawzhin Hozhabr Pour

Abstract

The benefits of analyzing driving behavior extend across various sectors, including insurance, transportation planning, and autonomous vehicle development. Insurance companies can customize policies based on individual risk profiles, promoting safer driving habits. In fleet management, the analysis assists in risk control, regulatory compliance, and enhancing customer satisfaction. Additionally, it plays a pivotal role in detecting and preventing accidents, contributing to safer road environments.

Over the past decades, significant advancements in Machine Learning (ML) techniques, particularly in learning relevant features for abstracted data representation, have been observed. The emergence of deep learning, utilizing Deep Neural Networks (DNNs), has further accelerated this trend, showcasing remarkable performance with ample data. This intersection has unlocked new possibilities for studying driving behavior, with ML playing a crucial role in extracting valuable insights from extensive driving data sets.

However, applying ML in this domain presents challenges. This outcome can be attributed to a variety of influencing factors. Driving involves a complex blend of cognitive, psychomotor, and perceptual activities that are hard to quantify and model accurately. Therefore, in this work, in-car sensor data is employed, as it is cost-efficient, widely accessible, and provides access to comprehensive vehicle parameters (e.g., speed and acceleration) with real-time data precision.
Driving behavior is inherently subjective, exhibiting significant variability among individuals and even within the same individual under different circumstances, which makes the labelling of data difficult and unreliable. To address this problem, this study leverages both supervised and unsupervised machine learning approaches and DNNs to detect all possible abnormal driving patterns in naturalistic driving patterns.
Capturing comprehensive real-world driving data that can reflect the full range of these variables is a massive, if not impossible, task. Detailed recording of individual driving behaviors can raise significant privacy concerns, and a true ground truth of dangerous driving behavior raises ethical considerations. This work presents a naturalistic driving data set (performed by drivers, spanning basic to professional skill levels) carefully assembled and supervised by experienced driving instructors. This data set encompasses annotated hazardous driving patterns derived from in-car sensors, which mitigates privacy concerns inherent in radar or visually-based data modalities.

Imbalanced data sets, a lack of positive (anomalous) samples, and interpretability issues in complex ML models further complicate the landscape. Therefore, comprehensive feature extraction methods using ML and DNNs are employed in this work to detect accidents within a naturalistic data set.

This work aims to address these challenges and gaps in the literature, focusing on anomaly detection and event detection in driving behavior analysis. The investigation revolves around two categories of questions:

1. Utilizing primary in-car sensors using ML approaches:

- Is it feasible to use primary in-car sensors with ML approaches to detect abnormal driving patterns?
- Is there a benefit in employing unsupervised deep learning models for anomaly detection compared to traditional ML models?
- Can the proposed solution be applied to a benchmark driving data set effectively, considering the lack of labeled driving patterns?

2. Detecting real-world accidents based on primary in-car sensor data:

- Is it possible to detect real-world accidents using primary in-car sensor data?
- What is the best feature extraction method for accident detection, and which features contribute significantly to the classification result?

Each of these questions is examined in detail and has led to new insights, using advanced machine learning techniques to manage the complexity of detecting abnormal driving behaviour, including accidents.

Chapter 2 unfolds into three significant sections, each offering valuable insights into anomaly detection in driving patterns. The initial segment introduces a foundational PRC framework for anomaly detection, achieving outstanding performance with supervised k-Nearest-Neighbors (kNN) and impressive results with unsupervised Gaussian Mixture Model (GMM). The second section delves into unsupervised anomaly detection using the proposed PRC framework on unlabeled Controller Area Network Bus (CAN-Bus) signal values, emphasizing the effectiveness of Autoencoders (AEs), particularly the noteworthy Long-Short-Term Memory Autoen coders (LSTM-AE), in detecting anomalous driving patterns based on speed and brake signals. The final section presents a benchmarking data set of naturalistic hazardous driving behavior, yielding remarkable results with Handcrafted Features (HC) classified by Support Vector Machine (SVM).

In chapter 3, a framework for accident detection using in-car sensors from a naturalistic data set is presented, employing diverse ML approaches. Notably, the combination of Convolutional Neural Network (CNN) features and an SVM classifier stands out, achieving impressive accuracy and showcasing promising performance given the reliance on naturalistic accidents and the limited samples

of severe accidents recognized by only four basic in-car sensors. Interpretability studies reveal the complementary nature of traditional feature engineering and DNNs in extracting optimal features from different sensor channels, enhancing overall effectiveness.

Despite challenges in data acquisition and dealing with imbalanced data, this work significantly advances exploring and benchmarking various anomalies and accident detection in naturalistic driving data sets. Outstanding results indicate the strong potential for driving behavior analysis using ML and DNNs utilizing in-car sensor data.

Zusammenfassung

Aus der Analyse des Fahrverhaltens ergeben sich Vorteile in verschiedenen Bereichen, darunter im Versicherungswesen, in der Verkehrsplanung sowie bei der Entwicklung autonomer Fahrzeuge. Versicherungsunternehmen können ihre Policen auf der Grundlage individueller Risikoprofile anpassen und so sicherere Fahrgewohnheiten fördern. Im Flottenmanagement hilft die Analyse bei der Risikosteuerung, der Einhaltung regulatorischer Anforderungen und der Verbesserung der Kundenzufriedenheit. Darüber hinaus spielt sie eine entscheidende Rolle bei der Erkennung und Vermeidung von Unfällen und trägt so zu einer sichereren Straßenumgebung bei.

In den letzten Jahrzehnten wurden erhebliche Fortschritte bei den Techniken des maschinellen Lernens (ML), insbesondere beim Erlernen relevanter Merkmale zur abstrakten Datenrepräsentation, erzielt. Das Aufkommen von Deep Learning unter Verwendung von Deep Neural Networks (DNNs) hat diesen Trend weiter beschleunigt und zeigt eine bemerkenswerte Leistung bei Vorliegen grosser Datenmengen. An diesem Schnittpunkt eröffnen sich neue Möglichkeiten für die Untersuchung des Fahrverhaltens, wobei ML eine entscheidende Rolle bei der Gewinnung wertvoller Erkenntnisse aus umfangreichen Fahrdatensätzen spielt.

Die Anwendung von ML in diesem Bereich ist jedoch eine Herausforderung. Dieses Ergebnis lässt sich auf eine Vielzahl von Einflussfaktoren zurückführen. Autofahren ist eine komplexe Mischung aus kognitiven, psychomotorischen und wahrnehmungsbezogenen Aktivitäten, die schwer zu quantifizieren und genau zu modellieren sind. Deshalb werden in dieser Arbeit Sensordaten aus dem Fahrzeug verwendet, da sie günstig und leicht zugänglich sind und Zugang zu umfassenden Fahrzeugparametern (z. B. Geschwindigkeit und Beschleunigung) mit hoher Zeitauflösung bieten. Das Fahrverhalten ist von Natur aus subjektiv und unterliegt erheblichen Schwankungen zwischen einzelnen Personen und sogar innerhalb derselben Person unter verschiedenen Umständen, was das Labeling der Daten schwierig und unzuverlässig macht.
Um dieses Problem zu lösen, nutzt diese Studie ein sowohl überwachte als auch unüberwachte maschinelle Lernansätze und DNNs, um alle möglichen abnormalen Fahrmuster in echten Fahrsituationen zu erkennen.
Die Erfassung umfassender realer Fahrdaten, die das gesamte Spektrum dieser Variablen widerspiegeln können, ist eine gewaltige, wenn nicht gar unmögliche Aufgabe. Die detaillierte Aufzeichnung des individuellen Fahrverhaltens kann erhebliche Bedenken hinsichtlich des Datenschutzes aufwerfen, und eine echte Basiswahrheit über gefährliches Fahrverhalten wirft ethische Fragen auf. In dieser Arbeit wird ein realistischer Fahrdatensatz vorgestellt (von Fahrern mit einfachen bis hin zu profes-

sionellen Fähigkeiten), der von erfahrenen Fahrlehrern sorgfältig zusammengestellt und überwacht wurde.
Dieser Datensatz umfasst kommentierte gefährliche Fahrmuster, die von fahrzeuginternen Sensoren abgeleitet wurden, was die Bedenken hinsichtlich des Datenschutzes, die bei Radar- oder visuellen Datenmodalitäten bestehen, ausräumt.
Unausgewogene Datensätze, ein Mangel an positiven (anomalen) Mustern und Probleme bei der Interpretierbarkeit komplexer ML-Modelle erschweren die Situation zusätzlich. Daher werden in dieser Arbeit umfassende Methoden zur Merkmalsextraktion mit ML und DNNs eingesetzt, um Unfälle in einem echten Datensatz zu erkennen.

Diese Arbeit zielt darauf ab, diese Herausforderungen und Lücken in der Literatur zu adressieren, wobei der Schwerpunkt auf der Erkennung von Anomalien und Ereignissen in der Fahrverhaltensanalyse liegt. Dabei werden Forschungsfragen aus zwei Bereichen untersucht:

1. Nutzung der primären Sensoren im Fahrzeug durch ML-Ansätze:

- Ist es möglich, primäre fahrzeuginterne Sensoren mit ML-Ansätzen zu nutzen, um abnormale Fahrmuster zu erkennen?

- Gibt es einen Vorteil beim Einsatz von unüberwachten Deep-Learning-Modellen für die Erkennung von Anomalien im Vergleich zu herkömmlichen ML-Modellen?

- Kann die vorgeschlagene Lösung effektiv auf einen Benchmark-Fahrdatensatz angewendet werden, wenn man bedenkt, dass es keine markierten Fahrmuster gibt?

2. Erkennung von realen Unfällen auf der Grundlage von primären Sensordaten im Fahrzeug:

- Ist es möglich, reale Unfälle anhand von primären Sensordaten im Auto zu erkennen?

- Welches ist die beste Methode zur Merkmalsextraktion für die Unfallerkennung, und welche Merkmale tragen wesentlich zum Klassifikationsergebnis bei?

Jede dieser Fragen wird detailliert untersucht und führt zu neuen Erkenntnissen, wobei moderne Techniken des maschinellen Lernens eingesetzt werden, um die Komplexität der Erkennung ungewöhnlichen Fahrverhaltens einschließlich Unfällen zu beherrschen.

Kapitel 2 gliedert sich in drei Abschnitte, die jeweils Einblicke in die Erkennung von Anomalien in Fahrmustern bieten. Im ersten Abschnitt wird ein grundlegendes PRC-Framework für die Erkennung von Anomalien vorgestellt, das mit dem überwachten k-Nächste-Nachbarn-Modell (kNN) eine hervorragende Leistung und mit dem unüberwachten Gaussian Mixture Modell (GMM) sehr gute Ergebnisse erzielt. Der zweite Abschnitt befasst sich mit der unüberwachten Erkennung

von Anomalien unter Verwendung des entwickelten PRC-Frameworks auf nicht gekennzeichneten CAN-Bus-Signalwerten, wobei die Effektivität von Autoencodern (AEs), insbesondere der Long-Short-Term-Memory-Autoencoder (LSTM-AE), bei der Erkennung anomaler Fahrmuster auf der Grundlage von Geschwindigkeits- und Bremssignalen hervorgehoben wird. Im letzten Abschnitt wird ein Benchmarking-Datensatz mit echtem, gefährlichen Fahrverhalten vorgestellt, der bemerkenswerte Ergebnisse mit Handcrafted Features (HC), klassifiziert durch eine Support Vector Machine (SVM), liefert.

In Kapitel 3 wird ein Framework für die Unfallerkennung mit fahrzeugeigenen Sensoren aus einem Real World Datensatz vorgestellt, bei dem verschiedene ML-Ansätze zum Einsatz kommen. Insbesondere die Kombination von Merkmalen, die mit Hilfe eines CNN erlernt wurden, mit einem SVM Klassifikator sticht hervor. Dieser Ansatz erreicht eine beeindruckende Genauigkeit und zeigt eine vielversprechende Leistung, wenn man bedenkt, dass es sich um echte Unfälle und eine begrenzte Anzahl von schweren Unfällen im Datensatz handelt, die von nur vier grundlegenden Sensoren im Fahrzeug erkannt werden. Studien zur Interpretierbarkeit zeigen die Komplementarität von traditioneller Vorgehensweise und DNNs bei der Extraktion optimaler Merkmale aus verschiedenen Sensorkanälen, was die Gesamteffektivität erhöht.

Trotz der Herausforderungen bei der Datenerfassung und dem Umgang mit unausgewogenen Daten bringt diese Arbeit die Erforschung und das Benchmarking verschiedener Anomalien und die Unfallerkennung in echten Fahrdatensätzen erheblich voran. Herausragende Ergebnisse zeigen das große Potenzial für die Analyse des Fahrverhaltens mit ML und DNNs unter Verwendung von Sensordaten im Fahrzeug.

Table of contents

Chapter 1

Introduction

Driving behavior analysis is essential in enhancing road safety, optimizing transportation systems, and improving overall driving experiences. By examining the patterns and actions of drivers, it becomes possible to identify risky behaviors, potential hazards, and areas for improvement. This analysis helps identify factors such as aggressive driving, distracted driving, and drowsiness, which significantly contribute to road accidents. Through the development of information and communication technologies (ICT) such as like telematics, onboard sensors, and data analytics, driving behavior analysis enables a deeper understanding of driver actions, allowing for targeted interventions and proactive measures to mitigate risks. By promoting safer driving habits and identifying areas where driver training is necessary, this analysis reduces accidents, saves lives, and minimizes the economic and social costs associated with road incidents.
The benefits of driving behavior analysis extend beyond road safety to various sectors, including insurance, transportation planning, and autonomous vehicle development. Insurance companies can leverage driving behavior data to offer personalized policies based on individual risk profiles. This data-driven approach allows for fairer premiums and incentivizes safer driving habits among policyholders.

The benefits of driving behavior analysis in fleet management encompasses various aspects, including risk management, regulatory compliance, and customer satisfaction. By closely monitoring driving behaviors, fleet managers can identify high-risk drivers and take necessary actions to mitigate potential accidents and insurance claims. This reduces financial liabilities and promotes a safer work environment for drivers. Furthermore, driving behavior analysis aids in ensuring regulatory compliance by tracking factors such as speeding, seatbelt usage, and adherence to traffic laws. This helps fleet operators avoid penalties, maintain a positive reputation, and uphold legal obligations. Additionally, analyzing driving behavior data enables fleet managers to provide accurate and timely information to customers regarding delivery times, route deviations, and driver behavior. This enhances customer satisfaction and builds trust in the reliability and professionalism of the fleet management service.

The contribution of driving behavior analysis in accident detection and prevention

is invaluable. By integrating advanced technologies like onboard sensors, and artificial intelligence, real-time monitoring and analysis of driving behaviors can be performed. This enables the detection of risky behaviors such as tailgating, sudden lane changes, and drowsy driving, which can lead to accidents. Early identification of such behaviors allows for timely intervention, such as issuing warnings to drivers or triggering automatic braking systems in vehicles equipped with advanced driver assistance systems (ADAS). Furthermore, analyzing driving behavior data can assist in identifying patterns and trends related to specific types of accidents, enabling targeted measures to be implemented. These measures may include traffic engineering interventions, road signage improvements, or modifications to road layouts to enhance safety. Overall, driving behavior analysis is a crucial tool in accident detection and prevention, helping to reduce accidents, save lives, and create safer road environments for all users.

Machine learning plays a pivotal role in driving behavior analysis, offering the potential to extract valuable insights from vast amounts of data related to such behavior. However, utilizing machine learning techniques in this domain comes with its own set of challenges. One of the primary challenges is the need for a sufficient amount of labeled data. Training a machine learning model to analyze driving behavior accurately requires a substantial data set with labeled examples of driving behaviors such as speeding, lane changes, and distractions. Collecting and annotating such data can be time-consuming and expensive. Additionally, the labeled data must be diverse and representative of various driving scenarios and conditions to ensure the model's generalization ability. Another challenge is the complexity and variability of driving behavior itself. Driving involves various actions, interactions, and contextual factors that influence behavior. Developing accurate models that can effectively capture and analyze these complexities requires sophisticated machine learning algorithms and feature engineering techniques. It is essential to consider multiple factors, such as driver characteristics, road conditions, traffic patterns, and environmental factors, to understand driving behavior comprehensively.

Despite these challenges, using machine learning in driving behavior analysis offers immense potential for enhancing road safety, optimizing traffic management, and advancing autonomous driving technology. With advancements in data collection technologies, improved algorithms, and access to large-scale labeled data sets, machine learning approaches can continue to evolve and address these challenges, leading to more accurate and effective driving behavior analysis systems. This groundbreaking thesis delves into the uncharted domain of detection of anomalies in driving behavior, employing cutting-edge machine learning (ML) techniques to unravel this intricate challenge. The thesis introduces a novel benchmark data set meticulously crafted to capture the complexities of risky driving maneuvers in real-world scenarios. The effectiveness of the proposed ML methodologies is rigorously evaluated against this benchmark data set, demonstrating their remarkable precision in discerning hazardous driving patterns. Furthermore, the thesis introduces a novel Pattern Recognition Chain, exhibiting exceptional capability in accident detection

within naturalistic driving data sets. This innovative approach holds promise for revolutionizing road safety by enabling the real-time detection of hazardous driving patterns, thereby paving the way for proactive interventions to mitigate traffic accidents.

1.1 Motivation

This section delves into the highlighting motivations and contributions of this thesis. Section 1.1.1 provides a comprehensive exploration of the intricate challenges regarding anomaly detection and accident identification within the domain of driving behavior analysis. To vividly illustrate the complexities inherent in event detection by analyzing in-car sensor data, section 1.1.2 masterfully employs a practical example. Section 1.2 succinctly summarizes the significant contributions made by this thesis, showcasing how they effectively address the aforementioned challenges. To provide a clear road map, section 1.3 offers a comprehensive overview of the structure of this thesis.

1.1.1 Context and challenges

As in most research tasks, the most significant challenge of driving behavior analysis is the need for appropriate ground truth data modalities for anomaly detection and accident detection in driving patterns. The reason behind that can be attributed to several factors:

1. **Subjective interpretation:** Driving behavior is inherently subjective. What one person may consider aggressive driving, another might perceive as efficient driving. No universally agreed-upon standard for categorizing or measuring driving behaviors makes it challenging to establish a solid ground truth.

2. **Variability in human behavior:** Driving behaviors can vary significantly between individuals and even for the same individual under different circumstances. This variability makes it challenging to generate a universally applicable driving behavior model.

3. **Complexity and multidimensionality of driving behavior:**Driving involves a complex blend of cognitive, psychomotor, and perceptual activities, which are hard to quantify and model accurately. This further complicates the establishment of concrete ground truth.

4. **Lack of comprehensive real-world data:** Driving happens in many conditions, situations, and environments. Capturing comprehensive real-world driving data that can reflect the full range of these variables is a massive, if not impossible, task.

5. **Privacy concerns:** Detailed recording of individual driving behaviors can raise significant privacy concerns, which could limit the availability and granularity of data needed to establish a ground truth.

6. **Ethical considerations:** To get a true ground truth of driving behavior, dangerous behaviors (like high-speed chases, drunk driving, etc.) would need to be included, which raises ethical considerations.

In the driving behavior analysis domain, several data modalities are commonly used to capture and analyze driving behaviors. These data modalities provide valuable insights into driver performance, safety, and behavior. However, they have their advantages and limitations. Telematics data, video data, external sensor data, smartphone-derived data, and traffic data are among the most well-known data modalities for driving behavior analysis. However, these modalities either need more detailed information regarding the detected events or are unreliable, inaccessible, or hard to set up. On the other hand, the benefits of ML-based approaches depend on the sufficient amount of labeled data that are difficult to access.

CAN-Bus (Controller Area Network) data is noteworthy and offers several advantages in driving behavior analysis. The CAN-Bus is a network that enables communication among various electronic components within a vehicle, providing access to a wide range of data related to vehicle performance and driver behavior. The following points underline the advantages and importance of using CAN-Bus data in driving behavior analysis:

1. Comprehensive Vehicle Insights: CAN-Bus data provides comprehensive insights into various vehicle parameters, such as speed, acceleration, braking, RPM (revolutions per minute), fuel consumption, and engine temperature. These data allows for a detailed analysis of driving behavior and performance, helping to identify patterns, deviations, and potential risks.

2. Accurate and Real-Time Data: CAN-Bus data offers high accuracy and real-time updates, capturing information directly from the vehicle's electronic systems. This ensures that the driving behavior analysis is based on precise and up-to-date data, enhancing the reliability of the findings.

3. Wide Accessibility: Most vehicles have a CAN-Bus system, that makes the data widely accessible. This enables driving behavior analysis to be conducted across various vehicles, allowing for comparisons, benchmarking, and generalization of results.

4. Objective and Quantifiable Metrics: CAN-Bus data provides objective and quantifiable metrics for driving behavior analysis. Parameters like acceleration rates, braking intensity, and RPM can be measured precisely, enabling the development of standardized metrics for evaluating driver performance and safety.

5. Cost-Efficiency: Utilizing CAN-Bus data for driving behavior analysis is cost-effective as it eliminates the need for additional sensors or equipment. The data is readily available within the vehicle's electronic systems, reducing implementation costs and complexity.

6. Potential for Advanced Analysis: CAN-Bus data can be combined with other data modalities, such as video or smartphone, to enrich the analysis. Integrating multiple data sources allows for a more comprehensive understanding of driving behaviors, their context, and potential correlations.

7. Diagnostic Capabilities: Besides driving behavior analysis, CAN-Bus data can also be used for vehicle diagnostics. It helps identify mechanical or electrical issues, maintenance requirements, and potential faults, ensuring vehicle safety and reliability.

Researchers, fleet managers, and safety organizations can gain valuable insights into driver behaviors, vehicle performance, and overall road safety by leveraging CAN-bus data in driving behavior analysis. The importance of CAN-Bus data lies in its ability to provide accurate, real-time, and comprehensive information, enabling data-driven decision-making and interventions to enhance driver safety and optimize driving behaviors.

Nevertheless, data acquisition is one of many challenges of the research tasks in driving behavior analysis. Identifying normal behaviors and deviating abnormalities are the biggest problems requiring sufficient normal and anomalous driving examples for anomaly detection. This process requires domain knowledge to understand the underlying patterns, especially dealing with unlabeled data. Evaluating anomaly detection performance using unsupervised ML models is an additional challenge to be solved.

Interpreting the decisions made by machine learning models is essential for trust and accountability. However, many complex machine learning models lack interpretability, making understanding why a particular prediction or anomaly detection was made challenging.

The first problem in case of accident detection lies in the nature of the accident event itself. A clear definition of an accident is required to be labeled and recognized. In most cases, distinguishing accidents with low speed, such as a slight contact of the car with the traffic barriers, from normal driving is hard. In general, annotating data for anomalies and accidents is a complex and subjective task. Experts must label instances correctly, which can be time-consuming and prone to errors. Moreover, agreement on what constitutes an anomaly or accident can vary across experts.

The second problem is the lack of positive (anomalous) samples. In real-world driving scenarios, accidents and anomalies are relatively rare compared to normal driving behavior. This leads to imbalanced data sets, where most instances represent normal behavior, making it difficult for machine learning models to learn patterns related to anomalies and accidents effectively.

Based on the aforementioned challenges and the limitations of the literature in anomaly detection and event detection in driving behavior analysis, this work is going to find the answers to the following questions:

- **Utilizing primary in-car sensors using ML approaches:** Is it possible to utilize primary in-car sensors using ML approaches to detect abnormal driving patterns? Considering the difficulty of adequately training and interpreting unsupervised deep learning models, is there a benefit to using them for anomaly detection as opposed to other traditional machine learning models? Considering the absence of annotated driving patterns, can the suggested solution be applied to a standardized driving data set?

- **Detecting real-world accidents based on primary in-car sensor data:** Is it possible to detect real-world accidents based on primary in-car sensor data? What is the best feature extraction method for accident detection? What features have more significant contribution to the classification result?

Before describing the contents of this thesis in more detail, the following subsection provides an example highlighting the difficulties of extracting in-car sensor data for driving behavior purposes. This example is taken from the LEICAR project in which the author of this thesis was involved.

1.1.2 Example : LEICAR - Learning-based multimodal interpretation of sensor data for event detection in car-sharing fleets

The following section describes an example illustrating the difficulties in extracting and utilizing in-car sensor data for the purpose of driving behavior analysis using ML algorithms. Experimental results reported in this section were carried out in the frame of the research project LEICAR (grant number: 01IS15048B), funded by the German Federal Ministry of Education and Research (BMBF) and reported in [1].

INVERS [2], the inventor of automated vehicle sharing, enables mobility operators to launch, run and scale with the first and market-leading Shared Mobility Operating System. INVERS supplies carsharing operators and company fleets with a complete modular system consisting of software and in-car technology. This technology enables the automated operation of a carsharing service or a shared pool of company vehicles. The central element of the Inverse system is a small electronic box installed in the vehicle. Connected to the vehicle via CAN, function-relevant information is read from the onboard network.
The collaboration of the University of Siegen and INVERS company aims to ease the procedure of recognizing primary in-car sensor data to enable further investigation in the domain of event detection in driving behaviors.
By recording all messages, based on the unique IDs for each message type, transmitted via the CAN-Bus it is possible to determine the associated signals with the timestamps.
However, these data cannot be automatically assigned to a substantial value, as it is not uncommon for several values to be combined in one single data part (e.g., the values of the individual wheel sensors). The way this division works can vary

greatly depending on the type of vehicle, and that's why we use different statistical methods to figure out which parts belong where.

Initially, no semantics are known for the messages transmitted on the CAN-Bus. i.e., the messages can only be understood as bit patterns. The data interpretation will occur in a multistage process developed during this collaboration.

Based on the brief introduction to the LEICAR project, the objectives of this project are in detail described as follows:

- In the first step, it is determined for the individual CAN message types which bits of a message belong together, i.e. form the binary representation of the value of a sensor signal. It is assumed that sensor signals are (essentially) continuous, i.e. they only change by a small amount from one message to the next. If one observes such a such a signal, the result is a statistical correlation between the course of the numerical value from the bits $k-1 \ldots 0$ with the course of the value of bit k. Therefore one can iteratively find all further bits of the signal value starting from one bit, provided that the signal changes over its entire value range. The latter is ensured by corresponding training sequences. In the second step, characteristic properties are determined from the signal curves found.
  The second step is to determine characteristic properties from which *features* for learning procedures can be derived. Examples for relevant properties of a signal course are the statistical distribution of the signal values, the statistical distribution of the differences of consecutive signal values, or the frequency spectrum of the signal waveform.

- With the help of the derived features, a classifier can now be trained, to assign a signal course to a certain type of sensor or a class of sensors and thus to determine or limit the semantics of the signal. This is possible because the signals of the different sensors have different distinguishable characteristics, as is illustrated in Figure 1.1 by means of four signal characteristics. During this procedure, additional sensors are searched to find additional correlations if there is still no clear assignment between a particular sensor (e.g., engine speed) and a signal waveform.

- Another important step for the reliable determination of the semantics of signal sequences is to check their mutual correlations, which are given by the physical properties of vehicles. For example, an acceleration curve must be (approximately) proportional to the derivative of the speed curve. Thus, if two signal curves are recognized as potential acceleration and speed and indicate the expected correlation, this significantly increases the reliability of the classification.

- To detect advanced events, e.g., driving style or (near-)accident, the semantically classified sensor signals can be fused and again assigned to a feature extraction. From this features, the events can be classified again by ML methods.

The following section addresses the problem of classifying sensor signals transmitted on the CAN-Bus inside cars. This work aims to solve the problem of dealing with large amounts of CAN-Bus data, finding the semantics of the most critical signals inside CAN-Bus in cars, independently of the car's model and manufacture. We aim to classify vehicle CAN-Bus data into primary sensor signals like speed, brake, steering-angle, and throttle that assume a prior role in finding driving behaviors.

Feature extraction and classification of sensor signals in cars based on a modified codebook approach

Due to the enormous demand for mobility and increasing road congestion, analyzing driving behaviors and preventing accident-prone situations has become a vital subject for scientific research. However, access to the car signals and their semantics should be provided before analyzing driving patterns.

In this context, we previously conducted a comprehensive analysis of driving patterns in order to develop an affordable, reliable, ecological, and environmentally compatible mobility solution within the LEICAR project (grant number: 01IS15048B), funded by the German Federal Ministry of Education and Research (BMBF). In this project, we had the opportunity to access to various car sensor values through CAN-Bus interfaces. We focused on four key signals for initial driving pattern analysis: brake, speed, steering-angle, and throttle. We established specific markers for most car models to filter out extraneous signal values. However, our goal of automated online event detection during driving required a model independent of car models and capable of accurately distinguishing signal semantics across various signal values. To achieve accurate signal classification, we proposed two feature extraction methods: Handcrafted Feature Selection (HFS) based on prior knowledge of distinctive signal features, and automated feature selection (AFS) based on the codebook approach.

Feature extraction

Before extracting features based on the vehicle type and individual sensors, we should note that our data set varies significantly in range, sampling rate, and type. Figure 1.1 shows four types of signals available in the data set. To address this problem, we needed to apply down-sampling and normalization on our data sets to extract the most efficient features, representing the characteristic of our signals. Therefore we used a sampling rate of 100 ms and zero-mean unit-variance normalization.

For the feature extraction of CAN-Bus signals in cars, we introduce two categories of handcrafted and automated methods, explained in the following subsections.

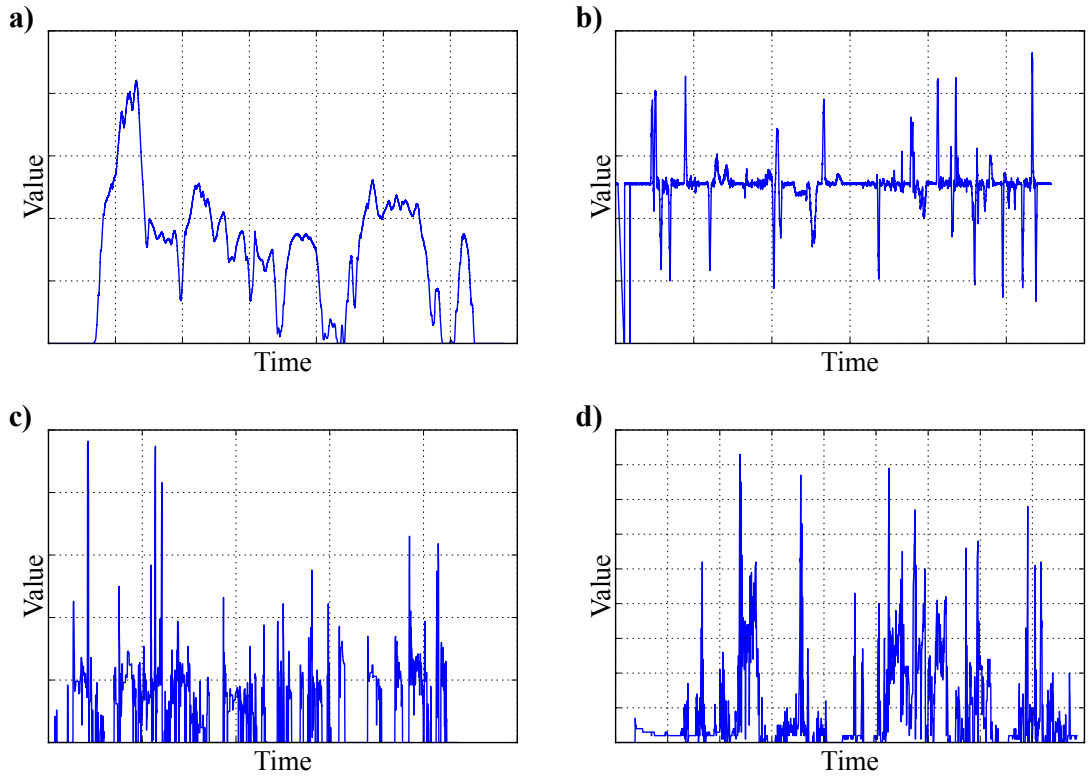


Figure 1.1: a) Speed, b) Steering-angle, c) Brake, d) Throttle.

Manual feature extraction

For the classification with manually selected features, we computed the following 14 feature values:

- The signal's first value (i.e., its value at the time 0) and the absolute difference of its first and last values are relative to the signal's maximum and minimum values. These two features provide some coarse information about the overall shape of the signal.
- A histogram of the values of the signal assumes using five equally sized bins between the signal's minimum and maximum values. This allows us to distinguish between signals that show clear peaks (e.g., brake) and signals that assume all values with relative probability (e.g., speed).
- The median of the signal's values relative to its minimum and maximum value. This should allow us to distinguish between, e.g., brake (just positive peaks, i.e., median close to the minimum) and steering-angle (peaks in both directions, i.e., median in the middle between minimum and maximum).
- The sum of the absolute differences of each signal value and the median value. This feature provides information about the variability of the signal.
- The signal's amplitude in five selected frequency bands (< 0.2 Hz, 0.2–0.5 Hz, 0.5–1 Hz, 1–5 Hz, > 5 Hz). The coarse spectrum helps to distinguish short peaks (e.g., brake) from smoother signals (e.g.throttle and speed).

All feature values are normalized into the interval $[-1, 1]$ before they are given to the SVM for classification.

AFS Based on the Codebook Approach

Unlike the common feature extraction representation, which is mainly based on prior knowledge of data, the codebook approach for emotion recognition yields a new representation of sequences, capturing remarkable differences among sensor data without considering prior knowledge.
Our codebook approach is based on [3] with modifications. In our case, using a different approach does not lead to desired results in finding distinguishable features since the range of sensor data collected from the CAN-Bus has a wide variety. On the other hand, compared to the handcrafted features, this approach has a long computation time.
Figure 1.2 illustrates an overview of our modified codebook approach. In this overview, the steps of the codebook-based feature extraction after having our data sets down-sampled and normalized are as follows:

- Divide the sequence of values in the signals into sub-sequences of so-called sub-windows with considering an overlapping factor.
- By using K-means clustering based on [4], group all subwindows into several clusters and set the cluster centers in a so-called codeword set.
- By assigning each sub-window to the most similar cluster center (using a soft assignment method) and representing them in a histogram with the frequency of codewords appearing in a signal, features of any specific signals are extracted. This histogram has the same dimension as the number of clusters and is considered a point in a multi-dimensional space.

Some remarks should be considered in implementing the codebook approach. The window-size w for dividing each signal into sub-windows, the overlapping size l, and the number of clusters should be chosen carefully. The codewords are high-frequency noises caused by a small w and l. In the last step of our approach, where each sub-windows is assigned to the most similar codeword, for flexibility, we have used a soft assignment based on Gaussian kernel density estimation from [5].
The frequency of each cluster center c_n in a given signal is represented as $F(c_n)$ formulated as follows:

$$F(c_n) = \frac{1}{S} \sum_{s=1}^{S} \frac{K_\sigma(D(x_s, c_n))}{\sum_{n'=1}^{N} K_\sigma(D(x_s, c_{n'}))} \tag{1.1}$$

where

$$K_\sigma(D(x_s, c_n)) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-D(x_s, c_n)^2}{2\sigma^2}\right) \tag{1.2}$$

Here $D(x_s, c_n)$ is the Euclidean distance. K_σ with smoothness parameter sigma is the Gaussian kernel density estimation.
In our modified codebook approach, the K-means algorithm based on paper [4] is extended by adding a well-known radial basis function (RBF) kernel [6].

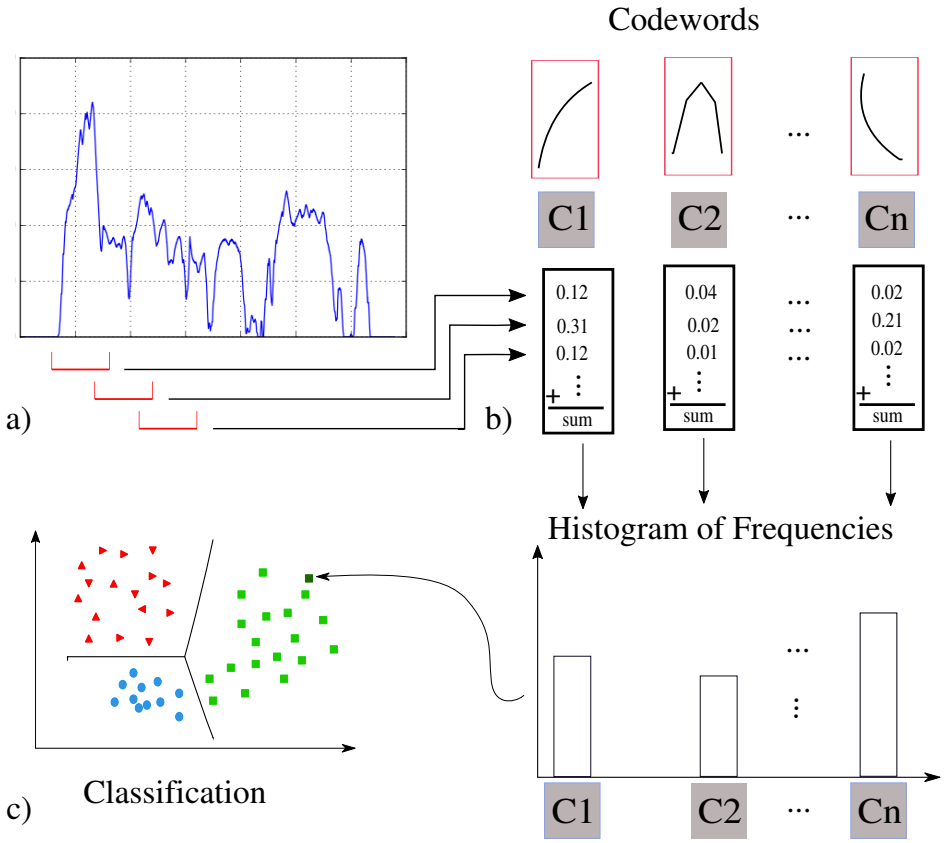


Figure 1.2: An illustration of modified codebook approach, a) and b) Codewords assignment, c) Classification training/test.

Classifier

By using handcrafted or codebook-based feature extraction, the aim is to classify sensor data received from CAN-Bus system inside cars. Thus, like other machine learning methods, we need to divide our data into training and testing sets. Afterward, we applied two main classification methods. The first is an SVM classifier based on linear and RBF kernels from papers [6] and [7].
Since the sequences captured from throttle and brake signals are very alike, we have decided to use not only one-vs-one classifiers but also a multi-class learning approach based on [8], which makes a significant improvement in our results.
For all of the classification tasks, the library implemented in [9] is used in our procedure, and finally, for tuning parameters in SVM classification, [10] has been applied to our application as well.

Results

Our data set is based on CAN-Bus traces from 30 rides with 13 car models. We have 230 sequences of in-car signals, and the objective was to detect signals of brake, speed, steering-angle, and throttle classes. The main parameters to be considered in our approach are window-size, overlapping factor, and number of clusters. In order to choose an appropriate window-size avoid capturing subtle noises and a significant

rise and/or fall in signal values, we have chosen a window-size range from 5 to 15 samples per sub-window with a sampling rate of 100 ms. A remarkable increase and/or decrease in signal values are visible within these ranges. The overlapping factor is chosen to be half of the window-size. We set the range from 5 to 10 clusters for the number of clusters in K-means clustering.
Concerning the importance of choosing the above parameters, in Figure1.3, two sets of codewords based on different window-size, overlapping factors, and clusters are illustrated.

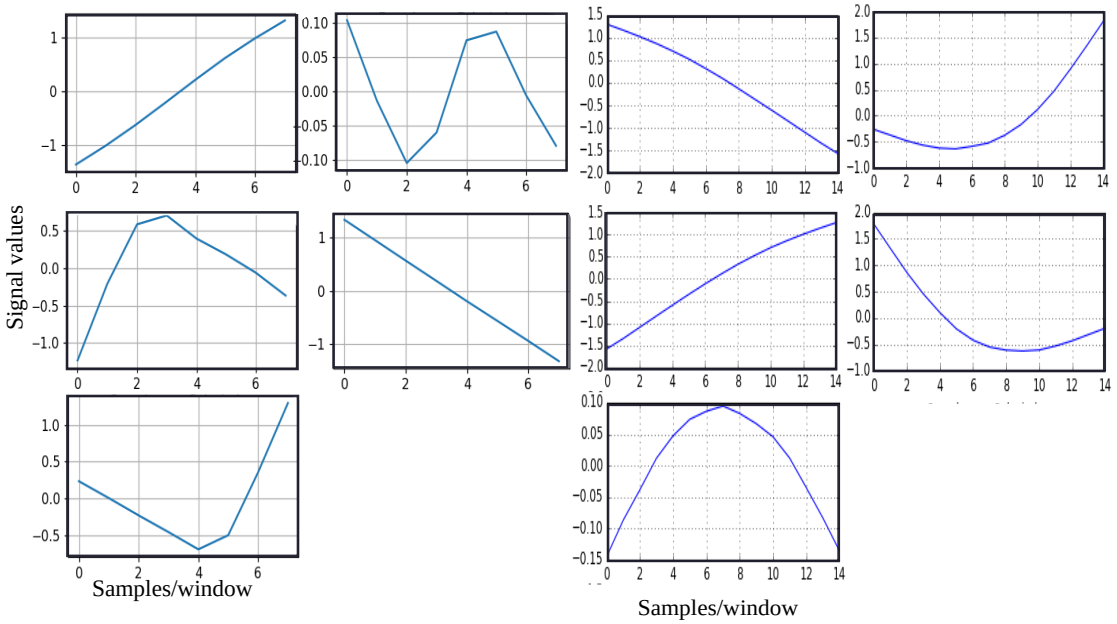


Figure 1.3: a) A set of codewords with window-size 8 and 5 number of clusters, b) A set with 15 and 5 number of clusters.

In Figure1.4, a bar chart of our results based on one-vs-one and multi-class classifiers, the classification score with different window-size is shown. Here we conclude that the bigger window-size with a multi-class classification achieves a significantly better result in our approach.

Conclusion and future work

Table 1.1 and Figure 1.3 demonstrate the result of different car signals' classification based on HFS and AFS. We demonstrated in our case that using handcrafted features gives a slightly better classification score of 93%. Despite achieving a higher classifier score based on HFS than the AFS, an efficient and accurate feature selection based on prior knowledge is labor and time intensive.
On the other hand, the classification result by choosing appropriate parameters based on AFS with the help of the codebook approach has gained a significant 90% score. Like nearly all automated approaches, the computation time of our modified codebook approach is noticeably high compared to the HFS.
After having most prior CAN-Bus signals classified independently of car models, our next plan is to detect sub-sequences within a trace of signal values where an event

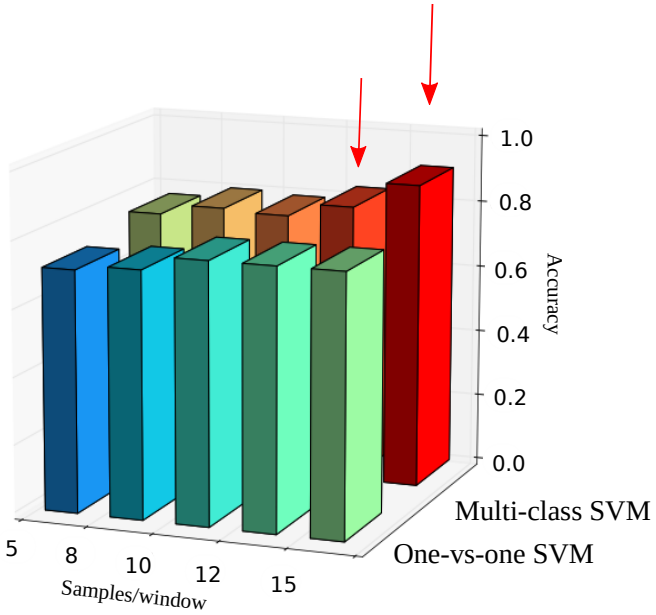


Figure 1.4: Accuracies of different classifiers based on AFS with different number of samples per window. A bigger window-size with a multi-class classification achieves a significantly better result.

Table 1.1: Performance comparison between AFS and HFS methods with different classifiers.

	Linear SVM Accuracy	Multi-Classification Accuracy
AFS	0.776	0.906
HFS	0.928	**0.934**

has happened. These events are time intervals containing abnormal signal value changes, mostly during unsafe driving. We are investing our research on car signals to point out any rare events during driving.

1.2 Thesis contributions

This thesis addresses the research questions mentioned in section 1.1.1 presented in papers [1, 11] and described in chapters 2 and 3. The scientific contributions of the presented work are as follows:

Anomaly detection in multimodal car sensor data: The first contribution of this thesis attempts to find a solution to tackle the various challenges in finding the most appropriate data modalities for analyzing driving pattern behaviors

and the lack of accessible and affordable ground truth of proper data modalities in driving behavior analysis.

In-car sensor data (e.g. CAN-Bus) have proven to fulfill the requirement of the desired data modalities due to many factors. Affordability and availability, in addition to the usability of these kinds of data independent from the vehicles model and the year of manufacture, are among the most characteristic of the in-car sensor data. Nevertheless, more research on ML-based anomaly detection in driving behavior needs to be done. The second chapter of this thesis presents an ML framework as a baseline for anomaly detection in driving patterns based on primary in-car data. Using this framework to provide a comparison study with supervised and unsupervised anomaly detection techniques. The novelty of this chapter continues introducing a benchmarking data set for anomaly detection based on car sensor data. Using the proposed framework to achieve promising results in driving anomaly detection with real-world driving data.

Automated car accident detection based on multimodal sensor data:
Accident detection is one of the most crucial subfields of anomaly detection in driving behavior analysis benefiting environmental or safety applications and the growing area of fleet management. Many works address the problem of accident detection by utilizing traffic data and external sensors (e.g., induction loop, acoustic sensors).

However, traffic data can be challenging to access, while external sensors can be difficult to set up and unreliable, depending on how they are used. Also, the need for accident detection data has limited the type of approaches used in the past, leaving, in particular, machine learning (ML) relatively unexplored. Thus, this chapter presents an ML framework for automated car accident detection based on multimodal in-car sensors using state-of-the-art feature extraction methods.

Chapter 3 of this thesis is the first study investigating ML-based accident detection on primary in-car network data. This study is distinctive and innovative research investigation on detecting natural driving accidents from the most accessible and affordable data sources inside cars. This chapter presents a detailed ML framework based on the PRC introduced to perform accident detection using primary in-car network data. It also uses this framework to compare state-of-the-art ML feature extraction techniques applicable to in-car sensor data for accident detection based on the SHRP2 NDS crash data set providing gas-pedal position, speed, steering-angle, and acceleration sensors. Promising results for automated accident detection based on a naturalistic data set are obtained using this framework.

1.3 Overview

This thesis is structured into two main chapters, each describing works carried out by the author of this thesis to address both questions introduced in section 1.1.1.

Chapter 2 describes the works to address the first question raised in this thesis about using primary in-car sensor data to detect anomalies in driving behaviors via ML approaches. This chapter proposes a Pattern Recognition Chain (PRC) to detect anomalies in unlabeled driving patterns. Furthermore, it provides a novel benchmark data set containing naturalistic hazardous driving patterns (with labels) and applies the proposed PRC framework with notable results.

Chapter 3 performs the first study investigating ML-based accident detection on primary in-car network data. It presents a detailed ML framework based on the PRC introduced in Figure 3.1 to detect accidents using basic in-car network data. It also uses this framework to compare state-of-the-art ML feature extraction techniques applicable to in-car sensor data for accident detection based on SHRP2 NDS crash data set providing gas-pedal position, speed, steering-angle, and acceleration sensors. Promising results for automated accident detection based on a naturalistic data set are obtained using this framework.
Finally, chapter 4 performs a summary of the findings reported in the frame of this thesis, in particular to which extent they addressed both questions raised in section 1.1.1 regarding the application of in-car sensor data for anomaly detection and accident detection in driving behavior. It concludes with an analysis of the work required to enhance and extend the proposed approaches and a discussion of future works.

Chapter 2

Anomaly detection in multimodal car sensor data

2.1 Problem statement

Anomaly detection has been a vital study focus of many research areas and application domains. Anomalous data usually contains important information about critical incidents. Anomaly detection in time-series aims to detect or predict from a set of training time-series whether a newly observed time-series is novel or standard. The detection of anomalies in driving patterns based on car data has many applications, e.g. detection of vehicle faults or monitoring of driving behaviors for the purpose of accident prevention. This chapter presents the rationale for investigating driving anomaly detection based on multimodal car sensors. In the following sections, the state of the art in driving behavior anomaly detection is discussed. Then, a PRC is presented to fill the gap in the literature on driving behavior anomaly detection based on time-series data in cars. The main steps of the proposed PRC framework during the implementation of this study are enhanced. Lastly, in the final section of this chapter, a benchmarking data set is launched in this domain of the research.

2.1.1 Motivation

Due to its significant influence on improving daily urban mobility, detecting anomalous driving patterns is an important research focus. Apart from transportation safety (such as advanced warning applications, e.g. Safety Awareness Services), monitoring anomalous driving patterns can be used by fleet companies. For instance, the quick dispatch of roadside assistance in case of vehicle breakdown or restricting the vehicle's usage in case of breaches of obligations (e.g. car racing, off-road) are a few contributions of anomaly detection in driving behavior used by fleet companies.

In addition to the aforementioned contributions of anomaly detection in driving behavior, the detection of abnormal driving patterns is crucial for the detection of accidents and near-accidents. Figure 2.1 illustrates the relationship between anomaly detection and accident detection in driving behavior analysis. If we

consider all the anomalous driving patterns that can appear in a driving data set, accidents are the events that occur least frequently compared to the usual (normal) driving events. Therefore, the ability to distinguish abnormal events from normal ones is a significant step towards recognizing certain abnormal events as accidents. By definition, an anomalous driving pattern does not match the normal expected driving pattern. Therefore, a simple approach to identifying anomalous driving behavior is to define a range representing normal patterns and consider any observation outside this normal range as an anomaly. However, most of the time, the boundaries representing these two classes are not easily distinguished, e.g. noise or outliers may be interpreted as an anomaly.

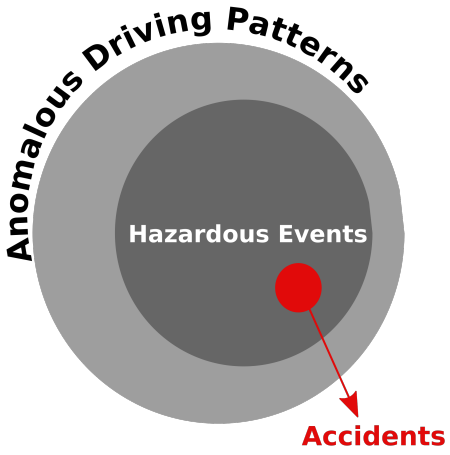


Figure 2.1: Exploring the interplay of anomaly detection and accident detection in the analysis of driving behavior.

Detecting anomalous driving patterns based on in-car sensor data is difficult to solve due to several factors. Outliers and noise are extraneous data lack meaningful information, impede data analysis, and contribute to biased results, while anomalies often contain significant and crucial information. Distinguishing between outliers and anomalies often requires extensive knowledge of the data. Depending on the nature of the data and the availability of labels, it is sometimes difficult to define normal and abnormal/anomalous patterns. Furthermore, the lack of labelled data and the scarcity and complexity of anomalous driving patterns are the reasons for the research gap in the field of driving anomaly detection based on in-car sensor data.
This study is a novel work investigating the detection of anomalies in driving behaviour using in-car sensor data. The following are the main contributions of this chapter:

- Providing an ML framework as a baseline for anomaly detection in driving

patterns based on primary in-car data. Using this framework to provide a comparison study with supervised and unsupervised anomaly detection techniques.

- Introducing a benchmarking data set for anomaly detection based on car sensor data. Using the proposed framework to achieve promising results in driving anomaly detection with real-world driving data.

The establishment of the aforementioned foundation depends on the completion of data generation. Consequently, this chapter is divided into three primary subsections aligned with the availability of the LEICAR project data set (see section 1.1.2). Prior to delving into the conducted experiments, the following section provides a concise overview of the PRC employed in this study.

2.1.2 Pattern recognition chain for anomaly detection in driving patterns

Pattern Recognition Chain (PRC) is a generic ML framework for solving classification tasks in various applications. It consists of discrete steps based on the focus of the applications. The choice of methods and the focus on the importance of the PRC steps make the different chains distinctive.
In this study, ML is used to detect anomalies in driving behavior by following a standard PRC framework (see Figure 2.3). The proposed framework for driving anomaly detection consists of five steps. Detailed information on each step is given in the following subsections.

Data acquisition:

Data acquisition is always one of the most crucial steps of the PRC procedure, which involves:

- Defining an experimental protocol for proper sensor setup
- Acquiring and merging data from appropriate sources
- Establishing a strategy for proper data labeling (in supervised learning approaches)
- Sampling the relevant sensor signals
- Converting the resulting samples to digital numerical values

The importance of data collection has several reasons. The first reason is the need for large data sets to train ML models properly. Next, the numerous following choices, such as the nature of the classes or the number of sensors and data acquisition channels, significantly influence the intricacy of the problem. These

choices include different operations depending on the ML task's context and purpose. The most important operations considered in our experiment are briefly described as follows:

- **Choice of classes:** The decision about the number of classes and their definition is context-dependent. In anomaly detection studies, the number of classes initially appears to be the two classes of normals and anomalies. In studies such as [12], where anomaly detection is based on hierarchical algorithms, the anomalous class is sorted into scored anomalies depending on the characteristic of the anomalous samples. Next, defining classes is a challenging operation which acquires a deep knowledge of the data and the class requirements. In the field of anomaly detection in driving patterns, the class definitions are not standardized in the literature (see section 2.2). Establishing class definitions can also be dependent on other parameters. For instance, in the domain of anomaly detection in driving behavior, defining classes and setting boundaries for class definitions are highly dependent on the implementation tools and options available to perform such experiments in simulated or natural environments.

- **Choice of sensors:** Similar to the class definitions, the choice of the sensors is highly based on the study tasks requirement and the characteristic of classes. In the domain of anomaly detection in driving behaviors, there are assorted data modalities such as statistical traffic data, camera-based or sensor-based (e.g. induction loops in traffic sensors, CAN-Bus signals) data modalities to capture anomalous driving patterns (see section 2.2). Based on the availability of such implementation setups and the study goal, one can choose sensors for data acquisition of anomaly detection in driving behavior.

- **Labeling strategy:** Labeling data in ML means annotating raw samples with identified information so the ML model can learn from it. Usually, labels are chosen as integers $\{1, 2, 3, ..., C\}$, where C refers to the number of classes. In time-series data, such as data used in this study, either a whole time-series of one recorded sample or the subsequences (windows) of a record is annotated with individual labels. In general, data labeling is mostly performed manually, which requires expert knowledge and usually takes time and requires labor. The quality and quantity of the labeled data play a critical role in the performance of the ML models.

Pre-processing:

Data pre-processing refers to operations that clean the data of flaws usually caused by transmission errors or sensor failures. In particular, they include operations such as eliminating any duplicates, or irregularities in the data, normalizing the data to compare, filling out missing data values, which is a commonly encountered problem, and providing the ML model data that are consistent. Data denoising is an essential procedure for anomaly detection studies. Noise can look remarkably like anomalies. The difference lies in the information behind it. Anomalies contain essential information that characterizes particular events, while noise (or outlier) is redundant or misleading information hindering the learning process. After denoising, normalizing and synchronizing time-series are the subsequent vital procedures of pre-processing step. Having different data from various sensors usually demands synchronization to match the time stamps of the recorded signals. Normalization techniques are used to unify time-series from different sensor channels to resolve this problem. Various normalization techniques have been utilized for ML pre-processing step. Though, The choice of pre-processing techniques is entirely dependent on the data set.

Segementation:

Usually, time-series raw data are recorded over a long period and might contain something other than homogeneous information. Most often, in the case of time-series anomaly detection, the anomalous event happens during a short interval of a long recorded sequence. Therefore, splitting the data into shorter discrete segments is needed to reveal the underlying properties of a special event. Sliding-window is a typical time-series segmentation approach which is most widely used. Figure 2.2 illustrates the sliding window approach. At first, a discrete time length -known as window-size, w-, based on the average duration of the specific events (which need to be classified) and their characteristic, is chosen. By sliding the window over the original sequence of time-series and creating time intervals, data segments of the chosen time length are created. In order to cover all possible creation of windows segments and extract as much information as possible from the original raw data, an overlapping factor l often corresponding to the 50% of the windows-size is considered to slide over already created windows.

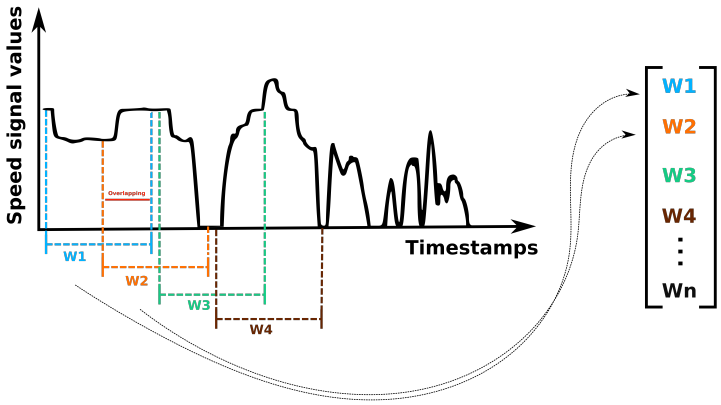


Figure 2.2: Sliding Window Approach (Windowing): Applying a sliding window technique to a speed signal sample, where W represents the window-size, and an overlapping factor is used.

It should be noted that in the case of supervised learning, the labeling procedure must be adjusted after segmentation. Initially, the labeling procedure is done based on the information about the corresponding time stamps indicating the events' starts and ends. Commonly, a whole window is annotated with the majority label of the time interval contained in the window-sized segment.

Feature extraction:

Feature extraction is the process of reducing the dimension of the raw data into an informative abstract representation of the data. In order to derive the most relevant information from the data, remove the redundant noise remaining after pre-processing and reduce a large amount of unnecessary information, the preprocessed data is replaced by its features and then fed into the model to be trained. Extracting features reduces the complexity of training a classifier. Suppose a classifier is trained based on detailed information that does not represent the desired event's high-level characteristics. In that case, the model can not be generalized to deal with unseen data. This so-called over-fitting can be minimized or prevented by appropriate feature extraction approaches.
The most commonly used feature extraction approach is the traditional feature engineering method consisting of handcrafted features that we refer to as HC in the following sections. HC features have been implemented for decades and still serve as a powerful tool when combined with ML classifiers. Traditionally, HC features are engineered based on expert knowledge of the data, which is only sometimes available. In this case, it is common to use simple statistical attributes computed on the time-series which have shown to perform well in practice despite their simplicity.

Feature learning is the second category of feature extraction algorithms discussed in this thesis. Feature learning is the automated procedure of learning features by Deep Learning approaches. The increasing popularity of Deep Learning makes

feature learning very appealing nowadays. Feature learning approaches are not used in this part of this dissertation. The reason is that the amount of data for the experiments in this section was insufficient for feature learning approaches. Therefore, this section briefly explains only the feature extraction techniques used for this experiment. The details of feature learning algorithms can be found in section 3.3.3.

Feature Extraction Based on Handcrafted Features (HC):

Traditional HC feature extraction has been implemented for decades, and due to its simple set-up, it still serves as a robust baseline when combined with ML classifiers. We computed simple statistical features commonly used in many application domains using time-series data for HC features. The HC features extract information from simple statistical attributes such as minimum, maximum or percentile or more elaborated descriptors such as features related to the frequency–domain based on the Fourier transform of signals. These features are calculated separately for each sensor channel. Extracted features from different sensor channels of each segment are concatenated to obtain a feature vector. The feature vectors and their corresponding labels are then fed to the classification model to be trained.

Classification:

Classification is the final step of the supervised ML framework to train a model to predict class labels (categories) $y \in \mathbf{Y} = \{1, 2, 3, ..., C\}$, where C is the number of classes, for the given feature vectors $\boldsymbol{x} \in \mathbf{X}$, $\mathbf{X} = \mathbf{R}^{F \times N}$, with F and N is the feature's space dimension and the number of data samples, respectively. There are two types of binary and multiclass classifiers. Anomaly detection classification problems usually use binary classifiers where labels $y \in \mathbf{Y} = \{0, 1\}$ contain two categories of normal and anomaly. Whereas multiclass classifiers involve assigning a data sample to one of several labels.

A large number of classifiers have been introduced in the past literature. The choice of the classifier depends on many factors, such as the classification task, the size, the quality and the nature of the data and the available computation time. Many classifiers aim at defining a linear function to separate classes. The most popular linear classifiers for anomaly detection are Support Vector Machine (SVM) (linear kernel) [13], Logistic Regression [14], and Naive Bayes [15]. Nonlinear classifiers are used to separate data instances which are not linearly separable. k-Nearest-Neighbors (kNN) [16], Random Forest (RF) [17], Multilayer Perceptrons [18] and Decision Tree [19] are among the most commonly used nonlinear classifiers for detecting anomalies in the literature. SVM can perform nonlinear classification using kernel functions. Kernel functions (e.g. Gaussian Radial Basis) map data into higher dimensions. Hence, data samples can be classified into separable classes using hyperplanes.

Neural networks have been used for anomaly detection as well. In order to detect anomalies, a neural network is first trained on the normal data. Second, each sample is tested by the model. If the network rejects the test input, it is an

anomaly. Autoencoder [20] and Replicator Neural Networks [21] are the most generally used anomaly detection techniques based on neural networks.
Finding an appropriate classifier depends strongly on the characteristics of the data. Various comparative studies have been conducted to analyze the performance of the classifiers [22, 23]. Accuracy, precision and recall are the most common performance metrics used to evaluate and compare classifiers' efficiencies. $F1$-score is a popular performance metric for classifiers, which is based on the so-called harmonic mean of precision and recall.

Clustering:

Clustering is an unsupervised learning technique used to find groups of similar instances in the data points. Clustering-based anomaly detection techniques are based on these assumptions: First, normal data samples lie close to their closest cluster centroid, and anomalous instances are far from this centroid. Second, normal data samples belong to large clusters, while anomalies lie in small and sparse clusters.
Several clustering-based anomaly detection approaches have been presented in the past literature. K-mean clustering [24] and Gaussian Mixture Model (GMM) [25] are among popular clustering-based approaches applied in anomaly detection literature. Similar to unsupervised ML algorithms, evaluating the performance clustering approaches is challenging. Most common clustering evaluations are based on experts' knowledge, where samples of the clusters are manually compared to an existing ground-truth. In case of availability of ground-truth labels (even for a small subset of data samples), F-1 measure can be applied to the precision and recall of pairs. In this way, the clustering result is not affected by the labels.

2.2 Related work

Anomaly detection in driving behavior analysis is critical in ensuring road safety and improving driver assistance systems. Several research efforts have been devoted to this area to develop effective techniques for detecting abnormal driving patterns and behaviors. Due to the importance of anomaly detection in data analysis, there has been much work in this research area. Chandola V. et al.[26] extensively survey anomaly detection techniques developed in machine learning and statistical domains. An extensive review of several approaches to novelty detection was given in [27].

Detecting anomalies and outliers in vehicle sensor data is a highly interesting topic, and analyzing the abnormal patterns might conclude to serious driving events such as dangerous driving behaviors or near crashes and crashes. Nowadays, preventing such events has been a major scientific concern due to the high increase in vehicle crashes resulting in fatalities and injured drivers. Therefore there are many researches and articles regarding driving studies and analysis. Anomaly detection in driving patterns can be utilized for different study purposes. Some works aim to distinguish between different driving styles like aggressive, inattentive, drunk, and "normal "or safe driving. If we consider any driving style as a set of

some atomic driver activities, driver distraction detection or drowsiness detection can be included in this category of driving analysis research [28, 29].

One prominent approach is using machine learning algorithms, such as supervised and unsupervised learning, to identify anomalies in driving data. Various features and data sources have been proposed, including vehicle sensor data, GPS data, and video recordings, to capture relevant driving behavior information. By leveraging machine learning techniques, these studies have achieved promising results in detecting abnormal events such as sudden lane changes, aggressive acceleration or braking, and illegal maneuvers. Zhang, H. et al.[30] provides a comprehensive overview of the existing techniques and methodologies used in anomaly detection for driving behavior analysis. it propose a SafeDrive approach for detecting driving anomalies. SafeDrive detects abnormal driving behaviors by employing a State Graph (SG) - a model based on both contextual relations between statuses of the same type of data, such as speed, at different timings, and correctional relations between statuses of different types of data, such as the vehicle revolutions per minute (RPM) and gear position - and then an online detection based on the comparison of the real-time driving data stream with the SG.

Furthermore, in [31], the authors propose an unsupervised anomaly detection method based on recurrent neural networks (RNNs). They use driving data, including vehicle sensor data and GPS information, to train the RNN model. The approach aims to capture temporal dependencies and detect abnormal driving patterns. Experimental results demonstrate the effectiveness of the proposed method in detecting various driving anomalies.

In some other research areas, as in [32] and [33], the aim is to warn the driver prior to an incident of the risk, which can be a distraction, aggression, drunk driving, or external risk originating from another vehicle or object. Alternatively, for instance, [34] the aim is to score a driver's performance on a "safety scale" based on driving style and driver behavior analysis and provide real-time information and feedback to assist in order to improve consciousness and promote safe driving.

Zhou, Y., et al. [35] proposes a context-aware anomaly detection approach for driving behavior analysis. The authors integrate contextual information into the anomaly detection process, including weather conditions, traffic patterns, and road infrastructure. They propose a novel feature representation method that captures driving behavior and contextual factors. Experimental results demonstrate the approach's effectiveness in improving anomaly detection accuracy in different driving scenarios.

In our work, we directly capture CAN-Bus data connected through OBD adapters to the car. After encoding the semantics of different signal IDs by statistical filtering approaches, the essential, required signals for driving behavior and situations analysis are filtered out. Firstly, we aim to detect abnormal driving patterns using in-car sensor data independent of the car model and manufacture. In our work,

we tackle the problem of detecting anomalies by proposing a PRC framework using ML approaches on in-car sensor data. For this purpose, we detect anomalies in vital car sensors such as speed and brake. Through an innovative way of utilizing ML approaches to find anomalies with unsupervised and supervised approaches, we ease the procedure of outlier detection in car sensor data. In addition, we provide a benchmark data set of naturalistic anomalous driving patterns and use our proposed framework to detect abnormal driving patterns.

2.3 Anomaly detection in driving patterns based on speed data

2.3.1 Context

The first step in building a PRC framework (described in section 2.1.2) for detecting anomalies in driving patterns is to establish a basic foundation based on the most common ML approaches and test the framework on a relatively simple data set. Following the standard PRC framework in Figure 2.3 for detecting anomaly patterns in cars' speed data at the beginning of LEICAR project, speed data was derived from the CAN-Bus network of daily driving patterns.

The main goal of this section is to establish the foundation of the proposed PRC framework, analyze the performance of state-of-the-art ML approaches on the acquired CAN-Bus data, and identify potential improvements for further investigation into anomaly detection in driving patterns using CAN-Bus data. The first step is creating a simple data set by repeatedly driving a car over speed bumps to mimic anomalies in wheel speed patterns. Classical ML-based approach are applied to the aforementioned data to distinguish two classes: normal and abnormal wheel speed. Based on our findings, we propose improvements for further investigations.

2.3.2 Methodology

In this section, we provide a concise overview of the methodologies employed for detecting anomalies in driving behavior using speed data. We applied traditional ML techniques to identify anomalies in the speed signals from twenty-one driving tracks, each manually labeled for reference. Normal behavior is characterized as driving on flat roads, while traversing over speed bumps is classified as abnormal behavior. Traditional ML approaches are used in this part of the study. There are two reasons for this choice. Since this is the first attempt to apply ML approaches to CAN-Bus data to analyze driving behavior, it is more prudent to keep the model as simple as possible and improve performance with more complex models once it succeeds. The second reason is that the amount of data at this stage is insufficient to train deep neural networks. A succinct description of the methods utilized is presented in the following.

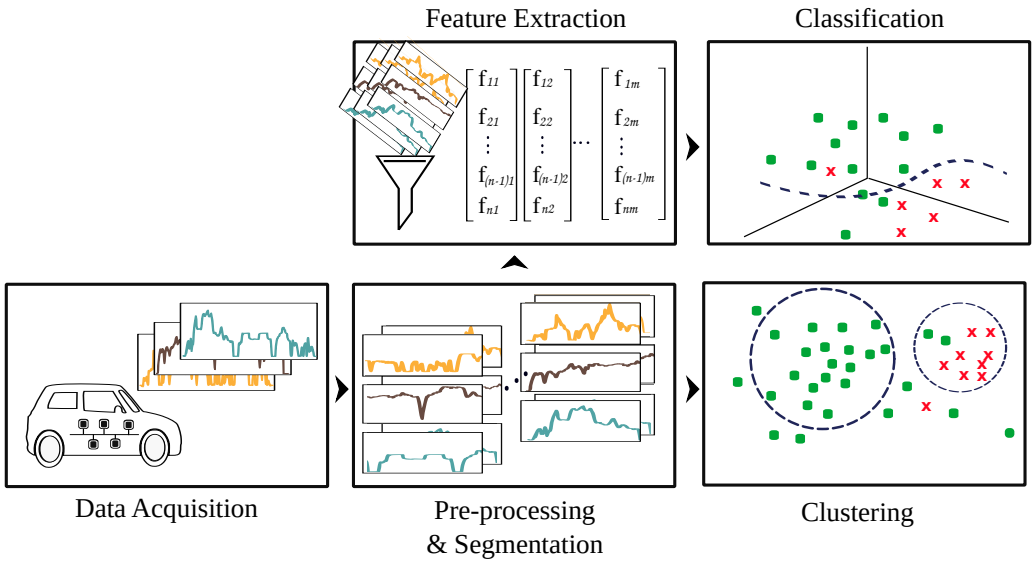


Figure 2.3: Pattern Recognition Chain (PRC) for anomaly detection in car sensor data based on traditional ML approaches.

Handcrafted feature extraction for anomaly detection in speed data

Features are extracted renditions of the data in a lower dimension carrying the essential attributes of the input data. The idea of training classifiers with the extracted features is to reduce the amount of redundant input information, which costs time and computation power. Feature engineering involves extracting and transforming variables from raw data so that features can be used for training and prediction. Feature engineering can require significant technical effort based on expert knowledge.

Handcrafted (HC) feature extraction is one of the most potent traditional feature engineering approaches used in classification problems for decades. Due to the satisfactory results, the extraction of handcrafted features for ML applications has been viral. In order to acquire the best knowledge of the data of the classification problem, one must access experts' knowledge of the desired data. However, it is only possible to reach experts' opinions in some cases. Therefore, with reasonably good performance, a standard solution is to use statistical attributes of the data, such as used in [1]. This experiment extracts nine HC features, listed in Table 2.1. For each window of segmented speed data, a feature vector of $\boldsymbol{x} \in \mathbf{R}^{F \times N}$ – with F and N being the feature's space dimension and the number of data samples, respectively– is based on concatenated listed statistical attributes extracted.

Classification

For the last step of the ML framework in this experiment, two supervised, and unsupervised classifiers are implemented to train and test the above-mentioned

Table 2.1: List of the HC features used on speed data.

Handcrafted Features		
Maximum	Minimum	Variance
Arithmetic mean	Integral	Largest gradient
Maximum Difference	Meancrossing	Second derivative

extracted HC features.

SVM: The first one is one-class SVM, a popular unsupervised classification approach in anomaly detection. SVM separates the data points via hyperplanes and maximizes the distance from them to the nearest data points from each side. If the data points are not linearly separable, this is where the kernel functions map the data to a higher dimensional space to be separable. One-class SVM was initially introduced by Schölkopf et al. [36] for novelty detection. One-class SVM maximizes the distance of the data points to the origin. This results in a binary function capturing the probability density of the data points with an output of +1 for the training data points and -1 for anomalies.

k-Nearest-Neighbors (kNN): The second classification approach used for this experiment is the k-nearest-neighbor classifier. kNN is a type of supervised learning algorithm used for regression and classification. kNN predicts the correct classes based on the distance of the tested sample to the k number of closest training data points. From calculated distances, instances can be classified directly, or probabilities of all class memberships can be determined. The classifier's results depend on the optimal choice of the value k. A large k prevents the effects of predicting the class by noise while making the class boundaries less distinctive. However, small k raises the probability of overfitting or misclassification.

Clustering

Clustering in ML refers to a process of arranging data points into groups (called clusters). It is an unsupervised ML approach, which groups the data based on the data points' similarities (in this case, the spatial distances) represented as vectors. Clustering for anomaly detection is based on two assumptions. First, normal data instances reflect the majority of instances contained in the data set. Second, normal data points are close to each other in the data distribution (based on the chosen distance metrics). Several clustering approaches are used for ML problems, such as centroid-, distribution- and hierarchical-based methods. In this experiment, K-mean [37], and Gaussian Mixture Model (GMM) [38] are used, which are centroid- and distribution-based clustering approaches, respectively.

K-mean clustering: K-mean clustering elects k cluster centers and continuously updates the cluster's center through iterative calculation. The calculations are based on minimizing the metric of the data points to their nearest cluster center.

Minimizing the metrics stops as certain convergence criteria are met.

Gaussian Mixture Model (GMM) clustering: The GMM is a distribution-based clustering approach that attempts to reconstruct the data distribution using the Gaussian model as best as possible. GMM assumes that the data set consists of several Gaussian distributions. As the distance from the center of a distribution increases, the probability that a data point belongs to it decreases.

The evaluation of clustering algorithms and unsupervised ML approaches is often subjective and difficult to interpret. If the data points were already classified into groups, clustering would be unnecessary. However, comparing the result of clustering with known classifications or the evaluation by a human expert is very beneficial in the case of labeled data. In addition, clustering is a suitable approach for data evaluation and processing in ML. Therefore, in this case, the standard F1-score is applied to the precision and recall for the clustering methods' evaluations.

2.3.3 Experiments and results

This section presents the implementation details of performed PRC framework, including the above-mentioned applied algorithms, the evaluation setting, and the results.

Data acquisition

According to the data acquisition procedure described in section 1.1.2, speed data are collected from *Ford Fiesta*. The vehicle is repeatedly driven over a speed bumper (as seen in Figure 2.4). Each driving duration varies by around one minute. Overall the data contain twenty-one driving traces of the CAN-Bus speed signal with a sampling frequency of 100 Hz. Speed signals from twenty-one driving tracks are manually labeled. Normal behavior is characterized as driving on flat roads, while traversing over speed bumps is considered as abnormal behavior.

Data pre-processing and segmentation

Before applying the proposed ML-based model, raw data have been preprocessed as follows. First, all segments and points that resemble the noise are discarded from further analysis. To address data bias arising from the structural characteristics of the data acquisition system, the denoising procedure was applied to specific time intervals of the CAN-Bus speed data, primarily at the beginning or end of the recorded time-series. Additionally, extended periods of vehicle inactivity were removed from the speed signals.

Then, the data were resampled to the sampling rates of 1/100, 1/200, and 1/300 ms. The min-max normalization was applied to convert the range of signal values to 0 and 1. In the next step, pre-processed data are segmented using the sliding-window technique. Various window-sizes from the range of $w \in \{2, 3, ..., 15\}$ are tested in

Figure 2.4: Speed data collection inside vehicles by driving over a bumper for the LEICAR project at the university of Siegen.

this experiment. After segmentation, the labeling procedure is manually adjusted according to the window-size. A subwindow is normal if no anomalous data point is stored in the associated subwindow. A subwindow is an anomaly if at least half of the stored segment contains anomalous behavior.

Anomaly detection in speed data using traditional ML approaches

The whole experiment is coded in Python using SciPy [39] and Scikit-learn [40]. In the context of the classification algorithms discussed in this work, there are parameters that are not trained by the training set, namely hyperparameters that significantly affect the performance of classifiers' data learning and decision-making. Grid search is a tuning technique attempting to find the best combination of parameters from the set of possible parameters space to outperform the classifiers' performance. There are two commonly used generic grid search techniques. GridsearchCV is the technique used to optimize the classifiers' hyperparameters in this study which exhaustively considers all possible combinations of parameters while RandomizedSearchCV samples a given number of candidates from the parameters space distribution [41].

An ML model aims to predict previously unseen data. Overfitting happens when the model is not generalized enough to predict unseen problems. In other words, the model is trained too well, which fits very accurately with the training data. Learning the hyperparameters of an ML model and testing it on the same data set causes overfitting. An overfitted model achieves perfect performance with seen data but would fail with yet-unseen data. A train-test split generates two non-identical independent data sets from a given data. A ratio of 80% training data to 20% test data is usually chosen. However, in cases where the data is small or the train and test sets do not have the same distribution, the train-test split is a possibility of high bias since we miss some information about the data which are not used for training. To compare the models' performance comprehensively and prevent

overfitting, the data are split into a train and test folds using K-fold cross-validation [42]. Cross-validation prevents such bias by ensuring every observation of the data appearing in both train and test sets. K-fold cross-validation randomly splits the data in k folds, then fits the data to k–1 folds and uses the remaining kth fold to test the model. This process is repeated till every kth fold serves as the test set. An average of the recorded scores will be the performance of the model. In our case, $k = 5$ is chosen.

For this experiment, three sampling rates are tested, which are 100 ms, 200 ms, and 300 ms/samples. Various window-sizes corresponding to the number of samples in a time frame are from $w = 2$ to $w = 15$ samples per window tested during the whole experiment. An overlapping factor l of fifty percent is considered.

The details of the experiments for each experimented algorithm are described as follows:

K-means:
For K-means clustering, a range of two to four clusters is tested in this experiment. The K-means algorithm is run ten times with different random centroids, and the maximum number of iterations of a single run is set to three-hundreds.

Gaussian Mixture Model (GMM):
Similar to the K-means clustering, two to four mixture models are considered for the GMM model. Each component has its own covariance matrix. The convergence threshold is set to 10^{-3}, and the regularization parameter of 10^{-6} is added to the covariance matrices. The model is one-hundred times iterated.

k-Nearest Neighbor (kNN):
Various numbers of neighbors from one to ten are considered for the kNN classifier in this experiment. A default range of parameter space (point-to-point radius) $radius = 1.0$ is set for finding neighbors.

Support Vector Machine (SVM):
For the unsupervised one-class SVM, with the help of grid search, various combinations of γ and ν are in the range of $\gamma \in \{0.01, 0.1, 1.0, 10, 100, 1000\}$ and $\nu \in \{0.1, 0.4, 0.7, 1.0\}$ experimented. The radial basis function kernel rbf is used for the kernel type.

Table 2.2 summarizes the experimented hyperparameters for the applied methods.

Results and analysis

The experimented methods are evaluated based on four metrics: accuracy, precision, recall, and $F1$- score. The results of the aforementioned approaches are provided in Table 2.3. The presented results are for the 1/300 ms sample rate, which achieved

Table 2.2: Hyperparameters of the applied models for anomaly detection using speed data.

Model	**Parameter**	**Value/ Type**
K-means	. # clusters	$\{\mathbf{2}, 3, 4\}$
	. # iterations for different centriods	10
	. maximum # iteration for each run	300
GMM	. # clusters	$\{2, 3, \mathbf{4}\}$
	. convergence threshold	10^{-3}
	. regularization parameter	10^{-6}
kNN	. # neighbors	$\{1:10\}\{\mathbf{4}\}$
	. point to point radius	1.0
SVM	. γ	$\{0.01, 0.1, 1.0, \mathbf{10}, 100, 1000\}$
	. ν	$\{\mathbf{0.1}, 0.4, 0.7, 1.0\}$
	. kernel function	rbf

Table 2.3: Evaluation metrics for the classification (in percentage) of anomalous speed signals using different models.

Methods	**Accuracy**	**Precision**	**Recall**	***F*-1 Score**
K-means	73.68	73.68	100	84.84
GMM	90.97	89.81	98.97	94.17
kNN-Unsupervised	82.66	87.85	87.85	87.85
kNN-Supervised	62.66	96.66	100	**98.03**
SVM	73.68	73.68	100	84.84

the best scores compared to other sample rates.

The main observations drawn from Table 2.3 are as follows. Firstly, supervised kNN outperforms other methods with a remarkable 98.3% $F1$-*score*. GMM model achieves the second-best result with a 94.17% $F1$-*score* which is a notable performance for an unsupervised clustering model compared to the best score with a supervised classifier. The third observation is the performance difference of roughly 10% between unsupervised and supervised kNN. The last important observation is the 100% recall score of three of the five models. To understand the reason for such high scores, one must closely examine the nature of the data and the pre-processing applied. As mentioned earlier, the results are based on the resampled speed signals with a sampling rate of 1/300 ms. The data is highly unbalanced and manually labeled. Any window containing an anomaly is considered an anomalous sample. At a sampling rate of 1/300 ms, the total number of subwindows decreases, but the relative number of anomalous subwindows increases. This change is due

to the fact that the probability of a subwindow covering an anomalous region increases when it has a larger spatial structure. Therefore, regions are classified as anomalies if they contain abnormal patterns but mostly have normal signal behavior.

In Figure 2.5, the performance of each models based on $F1$-*score* for the tested window-sizes w of the speed data with 1/300 ms sample-rate is illustrated. It can be observed that all models are achieving their best performances with window-sizes $w = 14$ and $w = 15$. Supervised kNN surpasses other models with the window-size of $w = 14$.

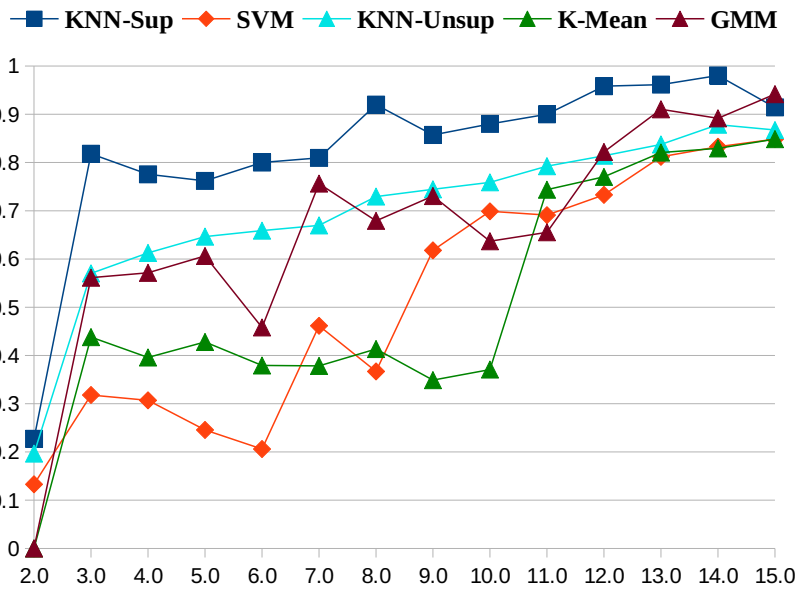


Figure 2.5: Model performances based on $F1$-*score* for different window-sizes. The x-axis shows the number of samples per window (w window-sizes), and the y-axis shows the models' performances in $F1$-*score*.

2.3.4 Conclusion

This section presents a baseline framework for anomaly detection in driving patterns using speed data from CAN-Bus signals. The following state-of-the-art ML algorithms have been evaluated, both supervised and unsupervised, namely kNN, SVM, K-mean Clustering and GMM. Based on the results and concerning the anomaly detection quality, the supervised kNN achieved the best performance with 98.03% F1-score. In the unsupervised case, the GMM algorithm obtains 94.17%, the second-best F1-score.

The high classification score of the supervised kNN is due to the learning with labeled data. These results could be further improved using a larger data set with more training examples. In addition, a more profound knowledge of anomalies' factual

occurrence could help train classification algorithms more effectively. Moreover, a manageable number of hyperparameters are tested using grid search. The optimal parameters could be further refined and the result optimized.
Another possible optimization technique can be used better to adjust the parameters, such as bayesian search [43] or black-box [44]. In principle, the experiments of the proposed framework presented in this section with the algorithms mentioned above can be widely applied for anomaly detection in sensor time-series data.
Alternatively, data from other CAN-Bus sensors recorded in parallel can be evaluated and used to investigate anomalous driving patterns. Furthermore, through the recording and analysis of data, statistical statements can be made about the frequency of risky driving situations and accident detection.

Based on the conclusion of the current observations, the stated ML-based anomaly detection framework will be in the following sections further explored. Additional CAN-Bus modalities and cutting-edge ML approaches, such as deep learning methods, will be examined.

2.4 Anomaly detection in driving patterns based on multimodal sensors from CAN-Bus data

2.4.1 Context

The previous section (see 2.3) establishes a baseline framework for anomaly detection in driving patterns using speed data from CAN-Bus. As the first attempt and as far as the quality and quantity of the acquired data allowed, the state-of-the-art ML approaches have achieved promising robust results so far. Nevertheless, various potential improvements exist to enhance and broaden the proposed framework.
Considering data acquisition as the first and most crucial step of any PRC framework, as is often the case in machine learning, the quantity and quality of the data are one of the more significant issues of the proposed PRC. In this regard, more driving data using additional sensor modalities (e.g., brake, gas, steering-angle) are being examined to improve and expand the proposed ML-based framework for anomaly detection in driving patterns. Enhanced deep learning approaches of anomaly detection techniques are then applied to the multimodal data, and the results are discussed in this section.

2.4.2 Methodology

This section briefly presents the method used to detect anomalies in driving behavior based on multimodal CAN-Bus data. The amount of acquired data in this state of the work enables utilizing an enhanced deep learning approach to expand the proposed PRC framework (see 2.3). Since the available data in this section is unlabeled, one of the most popular unsupervised ANN approaches is applied. A succinct description of the method utilized is presented in the following.

Anomaly detection using ANNs

The explosive success of ANNs in solving ML problems over the last decade has made ANNs also popular in the field of anomaly detection. ANNs have been applied to anomaly detection in multi-class as well as one-class problems. In order to apply anomaly detection using ANNs, at first, the model is trained on the normal training data. Second, each testing sample is fed into the model. The underlying assumption would be that an anomaly should be rejected by the model [45, 46]. Several ANNs have been applied in the domain of anomaly detection. Autoencoders (AEs) is one of the most popular ANNs, which are used for solving unsupervised anomaly detection study problems.

An autoencoder is an unsupervised neural network that seeks to learn a compressed version of an input. AEs are composed of an input layer and an output layer of identical dimensions, in addition to at least one hidden layer of significantly reduced dimension. Figure 3.6 illustrates a schematic representation of AEs. The most basic AEs have only one hidden layer. However, advanced AE approaches might contain multiple hidden layers.

AEs aims to produce an approximation of their input by first encoding the input data into lower dimensions and then decoding it to be as similar as possible to the original input. AEs consist of encoder and decoder, where the encoding of data happens on the way from input to the Hidden Layer(s) and decoding is done on the way to output. The idea behind using AEs for anomaly detection is based on the assumption that any anomalous input will result in a significant increase in the model's reconstruction error that the network provides as its output.
The concept of AEs is straightforward. However, like any simple approach, AEs have limitations, particularly in solving complicated tasks.
The main drawback of AEs is their use of Principal Component Analysis (PCA). PCA is an algorithm to reduce the dimensionality of the input features in the hidden layer(s)of AEs. PCA is a linear technique, so nonlinear attributes of the input data cannot be captured using AEs. Therefore, to learn the best from the input data using AEs, several advanced AEs are introduced in the literature. Among them, the most popular used in this experiment are as described as follows:

Feedforward AEs: Feedforward is the most basic implementation of the AEs, containing two layers for each encoder and decoder part. In order to obtain a high-level compression of the data, the encoding is set to increase the dimension of the input enormously.

Convolutional AEs: Convolutional Autoencoder (CAE)s employ the mathematical operation convolution ($*$), which in general continuous domain is defined as follows:

$$\mathbf{W} * \mathbf{I}_{x,y} = \sum_{dx=-a}^{a} \sum_{dy=-b}^{b} \mathbf{W}_{dx,dy} \cdot \mathbf{I}_{x+dx,y+dy} \tag{2.1}$$

where a and b are the size of the kernel W. The function id calculated for all x and y on the input data.
Convolutions are often used in image processing techniques. A convolution on time-series can be seen as applying and sliding a filter over a sequence of data or, in other words, as a generic non-linear transformation of an input vector I. AEs can be trained to decode(encode(I)) to learn the best filters for their input in order to extract relevant features in a compressed manner. CAE approaches apply convolution as either their primary encoding or in series with another feature modification. The common reason to use Convolutional Neural Networks (CNNs) is their ability to break down features and extract valuable information about their shape.

Long short term memory AEs: A Long short-term memory (LSTM) AE is an implementation of AEs using LSTM in the encoder-decoder architecture. LSTM is a variation of ANNs capable of learning temporal dependencies of sequences of input (such as time-series) as well as using an internal memory to remember and use information along the network. Utilizing LSTM in the architecture of AEs seems promising, as they can remember relationships during the learning process and can, therefore, recognize a better representation of the input. The capability of remembering more minor details and correlations in the data appears useful for anomaly detection purposes. Thus an LSTM AE is tested in this experiment. More information about the details of LSTM networks is provided in 3.3.3.

2.4.3 Experiments and results

This section presents the implementation details of the proposed ANN-based PRC framework for anomaly detection in driving patterns based on multimodal CAN-Bus signals, the evaluation setting, and the results. The experiments in this section of this study are written in Python with the help of SciPy [39], Scikit-learn [40] and Keras [47] libraries.

Data acquisition

The data set used in this part of the experiment is from the LEICAR, and the data collection setup is as described in 1.1.2. The recorded signals contain speed, brake, and steering-angle from seven car models, including BMW i8, Fiat Panda, Ford Fiesta, Hyundai IONIQ, Mercedes-Benz Vito, Toyota Auris and VW-up. The data set contains twenty-five trips with a duration of four to thirty minutes.

Data pre-processing and segmentation

To prepare the extracted speed, brake, and steering-angle data from CAN-Bus for anomaly detection based using ANNs few pre-processing techniques are applied. The procedure starts with removing undesired segments, which usually appear at the beginning of data recording. After cleaning, time-series data were resampled

at 1/50 ms (20 Hz). Then, normalization is performed to bring all signal values from different car models to the same range of min-max. In this step, the pre-processed data needs to be divided into segments of sequences that serve as input to the ANNs. Sliding-window technique with window-sizes $w \in \{10, 11, ..., 25\}$ and overlapping factors $l \in \{5, 6, ..., 13\}$ is applied on the data.

Anomaly detection in multimodal CAN-Bus data using Autoencoders

Before training the AEs, the data set is divided into training and testing folders with a ratio of 80 to 20%, respectively. A few hyperparameters for training AEs, such as batch size, epochs, negative sample ratio, and the classifier threshold, need to be optimized. Modifying these parameters can have a significant impact on the efficiency of the model.
Batch size is the number of training samples used for one iteration. The advantage of training data in batches, rather than feeding the entire training set into the model at once, is that the training procedure takes up less memory. This can be useful if there is not enough memory on the computer to train the data all at once. Also, by using mini-batches, the model is trained faster as the parameters of the model are optimized after each iteration. Careful choice of the size of mini-batches is crucial because they significantly affect the accuracy of the model.
An epoch is one complete pass of the training data set through the algorithm. Overfitting happens when the number of epochs in a model is more than necessary. In this way, the model learns the training data to a great extent, which fails to perform well on the testing set. To minimize overfitting and increase the generalization capacity of the model, the number of epochs should be optimal. To do so, usually, a part of the training data is assigned for validation of the model to check the performance of the model after each epoch of training. Different epochs from 5 to 20 are tested for each AE model, and the optimal number of epochs (20) is found manually.

The negative sample ratio is the following parameter to be considered for training the AEs. This is an estimated linear function for adjusting the model threshold concerning the approximated number of anomalies in the training data. For unlabeled data, knowing the ratio of negative samples is usually impossible. This implies that one must assume the percentage of anomalies in the training data for the algorithm to adjust its classifier threshold. If this value needs to be more satisfactory, it can still be easily adjusted manually after the model is built, as it is simply an estimated linear function.

The last adjustment before training AEs is choosing the models' activation functions. Generally, the activation functions used in AEs are nonlinear. Details of the structure of each AE model and their corresponding activation functions used in this experiment are outlined below. The structure of the hyperparameters is presented in Table 2.4. These parameters were selected based on the default options in the Keras model, manually adjusted, and tailored to the size of our data set

Feed-forward AE:

Table 2.4: Hyperparameters of the AE models on CAN-Bus data.

Model	Parameter	Value/ Type
Feed-forward AE	. # Encoder dense layers	2
	. # Decoder dense layers	2
	. # Neurons in layers	20
	. Activation function for the first layer	Tanh
	. Activation function for the second layer	ReLU
Convolutional AE	. # Conv. blocks	1
	. Conv. kernel size	(3, 1)
	. # Conv.kernels in each block	20
	. # Neurons in the dense layer	20
	. Activation function for the Conv. blocks	ReLU
	. Activation function for the dense layer	linear
LSTM AE	. # LSTM layers	2
	. # Output dimensions for each LSTM cell	128
	. # Neurons in the dense layer	20
	. Activation function for the dense layer	linear

Feed-forward AE is the most basic AE consisting of two layers for each encoder and decoder. The activation functions used in the feed-forward AE used in this experiment are hyperbolic tangent (Tanh) and Rectified Linear Unit (ReLU) for the first and the second layers of the encoder and decoder, respectively.

Convolutional AE:

The convolutional AE implemented in this experiment consists of a $1D$ convolutional layer followed by a 1D Global Max Pooling layer in addition to a Dense layer to fully connect the nodes. The activation functions used in this model are ReLU and regular linear activation for the convolutional and dense layers, respectively.

Long short term memory AE:

In this experiment, a basic LSTM AE is implemented with an LSTM layer of size 128 for each encoder and decoder, followed by a fully connected dense layer with a linear activation function. This network ensures that the input and output dimensions are matched.

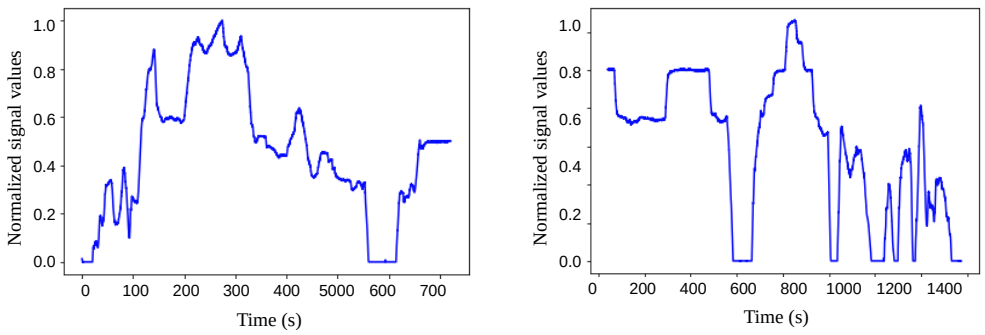
Results and analysis

In this section, the final evaluations of the above-mentioned algorithms are presented.

Before providing the results and analyzing the performance of the applied ANNs, two factors need to be considered. Firstly, the amount of data available for this study is larger than the data used in the section 2.3. The data includes long driving patterns, not just driving over bumpers, and the anomalies are unknown. Since manually labeling the data is impossible, the presented results are based on graph

interpretations and not performance metrics. Unsupervised clustering approaches are tested, and due to poor performance, the result is not presented here. So based on these facts, there is no mutual ground truth for comparing the results of this experiment to the previous section (section 2.3).

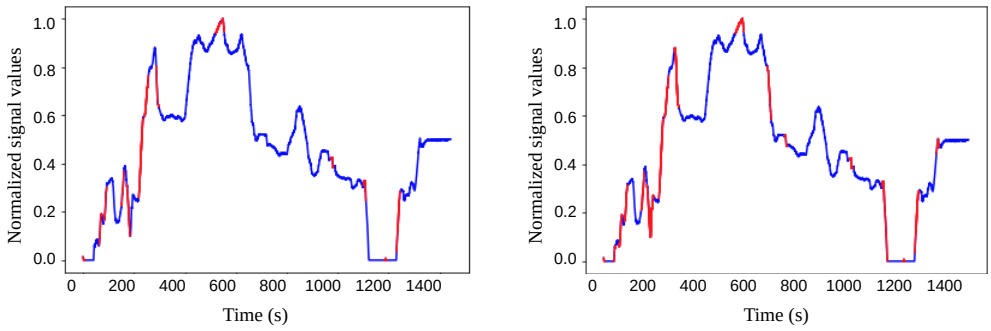
The first experiment is based on car speed data. The result of Feed-forward, convolutional, and LSTM AEs are similar. Figure 2.6a illustrates the speed data, and Figure 2.7b represents the detected anomalous segments with convolutional AE. The first observation in Figure 2.7b shows a promising performance of the model. Consecutive sharp decreases and increases within a specific distance are considered anomalies.
It should be noted that the detected anomalies are also very dependent on the choice of the threshold. Figure 2.7a represents the same model with a stricter threshold, detecting only the 25 most distant anomalies. 2.6b is another example of the CAE result on the speed data. All detected anomalies are presented in 2.8b, and the ten most distant anomalies are marked in 2.8a. An interesting result can be observed in 2.8b. Based on the recorded description of this trace and the marked anomalies in 2.8b standing at the traffic lights are also detected as anomalous samples. However, not all of the detected anomalies are accountable. There are samples identified as anomalies in which the speed changes by 1% to 2% of the maximum total velocity in 500 ms frames. Therefore, to better evaluate the performance of the trained AEs for anomaly detection in CAN-Bus data, a simple predictable speed sample from section 2.3.3 is tested. Figure 2.9a and 2.9b depicts the promising performance of AEs in detection anomalies in driving patterns based on CAN-Bus data.



a) Example 1: Speed data.

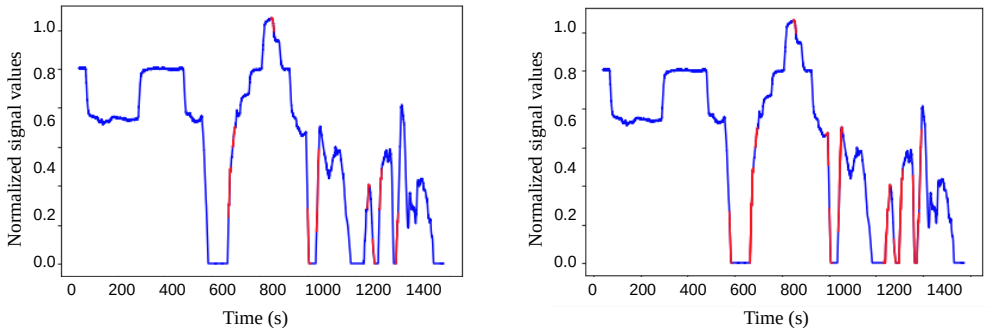
b) Example 2: Speed data.

Figure 2.6: Two examples of speed data used for anomaly detection approaches.



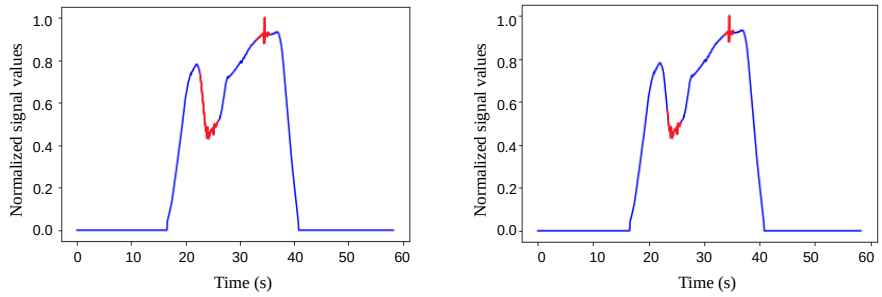
a) The 25 most distant anomalies. b) All detected anomalies.

Figure 2.7: Convolutional AE results of anomaly detection based on the speed signal.



a) The 10 most distant anomalies. b) All detected anomalies.

Figure 2.8: Convolutional AE results of anomaly detection based on the speed signal.

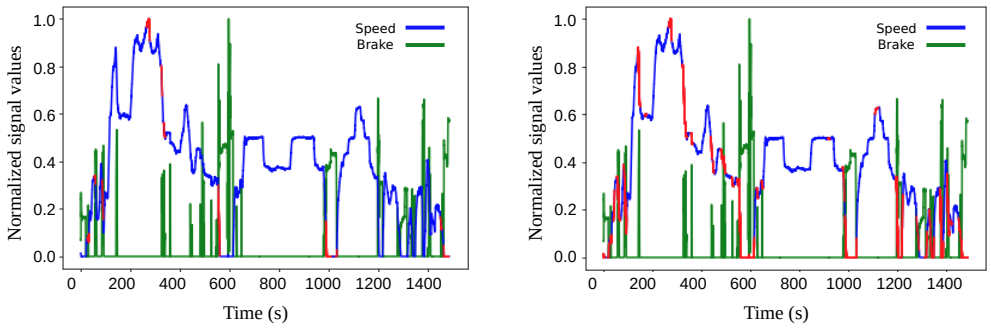


a) Example 1. b) Example 2.

Figure 2.9: Convolutional AE results of anomaly detection based on the predictable speed samples from section 2.3.3, where the car is driven over bumpers. a) Example 1 : Anomalies in speed data and b) Example 2 : Anomalies with restricted threshold.

In the next step, the brake signal as the second input modality is added to the experiment. The input of the AEs is concatenated speed and brake signals. In Figures 2.10a and 2.10b , the LSTM AE result shows the 50 most distant and all anomalies, respectively. As shown in Figures 2.10a and 2.10b, the speed values

are now significantly unaffecting. In this case, the solid consecutive oscillations in the brake signal determine the anomalies instead. It should be mentioned that, in general, LSTM AE performs noticeably better than other AEs dealing with frequent changes in speed signal.



a) The 50 most distant anomalies.

b) All detected anomalies.

Figure 2.10: LTSM AE results of anomaly detection based on speed and brake signals.

Based on all observations from illustrations, the overall result of the performed experiments are as follows:
AEs detect all sharp changes in speed data that appear as anomalies compared to the rest of the speed values. Not only 100% of the abrupt speed changes detected as anomalous samples, but some are also falsely classified. In this case, since the collected data is completely balanced and the model is not overfitted, ambiguity in the anomaly definition can be only caused by misclassification. The definition of what constitutes an anomaly is often subjective and application-dependent. Sometimes, anomalies may be data points that need to follow a simple, well-defined pattern, making it challenging for any model, including autoencoders, to consistently identify them correctly.In this study, variations of AEs are also tested, but almost no differences in the performance of different variations of AEs for anomaly detection in driving patterns based on speed data are observed.
The performance of AEs on the multi-modal CAN-Bus signals, including speed and brake, is more outstanding than a single modality input due to the abrupt changes in brake compared to speed values. In this case, LSTM AE outperforms other performed AEs by marking more accountable segments of the driving traces as anomalous samples. A smoothed speed signal using a Gaussian filter is also tested as an input, and no improvements in the performance are observed.

Despite the lack of accuracy metrics in this context, the promising performance of AEs for anomaly detection in time-series data can be observed based on the above-illustrated results.

2.4.4 Conclusion

In this section of the study pursuing the topic of anomaly detection in driving patterns, CAN-Bus signal values of twenty-five driving trips from various cars are analyzed according to the proposed PRC framework. Based on the state of the collected data (enough data for ANN models but unobtainable labels), different variations of AEs, one of the most popular unsupervised ANNs, are tested and analyzed.

According to the observed illustrations of the detected anomalies on speed data, all AE variations have a stable performance. The choice of the threshold impacts the final detection in all three variations of AE. The performance of AEs on the multi-modal CAN-Bus signals, including speed and brake, could have been more outstanding than a single modality. However, LSTM-AE outperforms other AEs in detecting anomalous driving patterns based on speed and brake signals.

To further analyze the performance of the before-mentioned AE approaches, a short overview of the collected data is necessary. Considering the underlying facts like the data used in this part of the study few crucial remarks should be noted. First, the recorded driving traces are normal everyday driving trips containing no dangerous events. Secondly, the data contain neither detailed information nor labels about the driving journeys. Therefore, AEs perform remarkably promising by detecting sharp speed acceleration (or deceleration) and standing behind the traffic lights as anomalous events in an uneventful daily driving trip.

Lastly, one of the most significant challenges related to unsupervised approaches is determining the accuracy. Since the processed data is not classified/labeled, and the objective of this work is to do so automatically, it is hard to determine the exact rate of correct classifications by the algorithm. Generating random noise is one proposed solutions, but mimicking dangerous situations with a mathematical model remains difficult to mimic. This work partly addresses the problem by using data with easily visible anomalies, although it will remain one of the most significant limitations.
Nevertheless, these results pushed the boundaries of unsupervised anomaly detection in vehicle sensor data to open the possibility for further research. The provided framework significantly impacts solving the problem of anomaly detection in driving patterns.

According to the above-performed study on anomaly detection in driving patterns based on multi-modal CAN-Bus data using ANNs and the underlying challenges regarded data limitations, the following points summarize the remarks on future work:

- Every single step of the pre-processing impacts anomaly detection. Data transformations that would inadvertently lead to compromising anomalies should be avoided in the pre-processing steps. This requires explicit knowledge of the data and the answers to the following questions. 1. What is an anomaly concerning the nature of the data set? 2. What are the characteristics of the

anomalous events we are seeking? 3. Can we define the anomalies directly, or should we look at how the machine learning model classifies them and base our definition on it?

- Despite the unpredictability of the data, unknown shape of anomalies, and their ever-changing range in the collected data set of driving trips, AEs show coherent and promising performance in finding anomalous driving patterns based on CAN-Bus signals. Thus, further investigation of using AEs for anomaly detection in more advanced data sets is recommended.
- Finally, the data's quality and quantity are prerequisites for a successful anomaly detection model. High-quality data are homogeneous, contain information describing the characteristics of the anomalies, and are preferably labeled. Therefore, for future work, it is crucial to prepare a collection of high-quality naturalistic data and reinvestigate the above-suggested framework that has shown promising performance in anomaly detection of driving patterns based on CAN-Bus signals far. In addition, Considering data collection for a multi-class anomaly detection model needs to be further analyzed in future work.

2.5 Anomaly detection in benchmark data

2.5.1 Context

Anomaly detection in driving behavior is a research domain that focuses on identifying abnormal driving patterns using various data modalities, such as traffic data and image or sound-based data modalities. In this work, internal car sensors are used for anomaly detection and event detection in driving patterns. The data collected from the sensors include information such as speed, acceleration, braking, steering-angle, and vehicle position.

Anomaly detection in driving behavior has become increasingly important in the research domain of recent years due to the rise of autonomous vehicles and the need to ensure the safety of drivers and passengers. An example of a contribution to anomaly detection in driving behavior is using machine learning algorithms to analyze sensor data and identify abnormal behavior patterns. These algorithms can be trained on large data sets of normal driving behavior to learn what normal behavior looks like and then use this knowledge to identify abnormal behavior. Another contribution is the development of systems that can provide real-time feedback to drivers when they exhibit abnormal behavior. These systems can significantly improve driving safety by helping drivers be more aware of their behavior and take corrective action when necessary. For example, a system may warn a driver if they are driving too close to another vehicle or exhibiting aggressive driving behavior.

The word anomaly can refer to various subjective events (car failure, sudden stops caused by traffic lights or traffic jams, etc.) in the concept of driving behavior.

In this study, anomalies are firstly defined as anomalous driving patterns that do not match the rest and appear less frequent than daily driving patterns. Based on this definition, any anomalous patterns such as a long stand behind traffic lights can be the outcome of the anomaly detection in driving patterns. The previous two subchapters present an ML-based framework with promising results (see subchapters 2.3.3 and 2.4.3) for anomaly detection in driving behavior (daily driving behaviors) using in-car sensor data.

However, the main purpose of anomaly detection in driving behavior is to identify potentially dangerous driving behavior, such as sudden stops, rapid accelerations, or swerving, and provide feedback to the driver to help prevent accidents. For this purpose, a data set containing dangerous driving patterns, preferably with labels, is vital. This sub-chapter introduces a naturalistic labeled benchmarking data set for anomaly detection in driving behavior. The methods with the most promising results mentioned in the last two sub-chapters are applied to this enhanced data set. Ultimately, the results are presented and interpreted.

2.5.2 Methodology

The methodology of this section is a continuation of the study on the proposed PRC framework for anomaly detection in driving behavior based on CAN-Bus signals. The objective of this section is to present a labeled data set of naturalistic hazardous driving behaviors to serve as a benchmark data set for detecting anomalies in driving patterns using primary sensor data in cars. Additionally, the data set will be used to evaluate the ML approaches used before.

Handcrafted feature extraction

For decades, handcrafted feature extraction has been a popular and effective approach for training machine learning models. It involves manually designing and selecting relevant features from the input data that are then used to train the model. These features are often selected based on expert knowledge or domain-specific understanding of the problem. For example, in image processing, handcrafted features include texture, color, and shape descriptors. Similarly, features such as pitch and frequency might be extracted from audio signals in speech recognition. Despite the success of handcrafted feature extraction, it can be time-consuming and requires significant domain expertise. Therefore, recent advancements in deep learning have focused on automating feature extraction to reduce the need for manual feature engineering.

Typically, the efficiency of the extracted information is optimized by an expert's knowledge of the relevance of the handcrafted features. However, due to insufficient resources, experts' knowledge is not always available. In such cases, low-level statistical attributes of the time-series data, such as minimums, maximums, standard deviations, etc., are often used in either the time or frequency domain [48].

Classification

SVM is one of the most popular classifiers in supervised ML applications. Based on the last observations in this chapter of this dissertation and the popularity of SVM in solving classification tasks in past ML works, SVM classifier is also used on the Lab data. SVM uses different kernels capable of classification tasks in higher dimensions, even in cases where the number of feature vector dimensions is higher than the number of samples. A soft-margin SVM classifier uses C parameter, which grants the ability to control how much the SVM algorithm penalizes the misclassification of the data points. Conversely, using a high value for C leads to a decision surface that classifies more points correctly, even at the cost of lousy generalization (overfitting) [49]. In this case, the chosen kernel for the applied classifier is the linear kernel. The linear kernel function is the most simple of the commonly used SVM kernels [50]. Models derived from the SVM approach described so far are, by definition, binary classifiers. An SVM is designed to find a hyperplane that serves as a decision boundary between two classes: normal and anomalies. However, in this part of the study, multiple classes will be introduced in the benchmarking data set. Therefore, it is necessary to differentiate between more than just two classes. Several approaches exist for multiclass classification using SVC in particular and binary classifiers in general. Generally speaking, these approaches entail constructing many classifiers, though there is also an approach to constructing an SVC using a multiclass objective function [51]. With our goal of constructing a classifier that can be run on embedded, in-car hardware in mind, computational complexity must be considered a decisive factor. This being the case, we will focus on the One-versus-all and All-versus-all strategies.

One-versus-all classifiers (OVA), also called one against rest or winner-take-all [52], is commonly cited as the simplest of binary classification decomposition approaches to multiclass classification. It consists of the following steps [53]:

1. Given K classes, construct K binary classifiers, where each classifier differentiates a given class from the other $K-1$ classes. Each classifier is trained considering training examples of its respective class as positive instances and training examples from one of the other $K-1$ classes as negative instances.

2. Classify unknown instances using each of the K classifiers. The instance is then considered to be of the label belonging to the classifier that produces the maximum output.

All-versus-all classifiers (AVA), also commonly known as one against one, is a slightly more sophisticated approach than OVA. According to [53], it works as follows:

1. Given K classes, construct $\frac{K(K-1)}{2}$ binary classifiers - one to distinguish between each pair of classes.

2. Classify unknown instances using each of the classifiers. Voting among all classifiers is then conducted. The label of the winning classifier is applied to the instance.

Performance analysis

Performance metrics are an essential component of evaluating the effectiveness of machine learning models. They provide a quantitative measure of a model's accuracy and ability to generalize to unseen data. Standard performance metrics in machine learning include accuracy, precision, recall, and F1-score.
Accuracy is the most basic performance metric that measures the proportion of correctly classified instances. Precision measures the proportion of true positives over the total number of predicted positives, while recall measures the proportion of true positives over the total number of actual positives. The F1-score is the harmonic mean of precision and recall, providing a balanced measure of a model's performance. In order to measure classification performance, selecting the right performance metric is critical to ensure the model's effectiveness and identify potential areas for improvement.
Precision: also called True Positive Accuracy (TPA) [54], is defined as the ratio of true positives (tp) to predicted positives (pp) [54]:

$$\text{Precision} = \text{P} = \text{tpa} = \frac{\text{tp}}{\text{pp}} \tag{2.2}$$

Recall: also known as True Positive Rate (TPR) [54], as the name suggests, is defined as the ratio of true positives (tp) to real positives (rp) [54]:

$$\text{Recall} = \text{R} = \text{tpr} = \frac{\text{tp}}{\text{rp}} \tag{2.3}$$

F1-score: The F1-score is defined as the harmonic mean of Precision (P) and Recall (R) [55]:

$$\text{F1} = \frac{2\text{PR}}{\text{P} + \text{R}} \tag{2.4}$$

Multi-class classification performance evaluation: The measures defined so far rely only on the concept of positives and negatives which translates well to an evaluation framework for binary classification algorithms, but falls short of providing a suitable metric for evaluating a Multi-class classification system.
A naive but also still commonly used approach that works for binary classifiers as well as multi-class classifiers is the accuracy measure. It is simply defined as the ratio of the number of correct predictions to the total number of predictions [56]:

$$\text{Accuracy} = \text{A} = \frac{\text{n}_{\text{correct}}}{\text{n}_{\text{total}}} \tag{2.5}$$

Though accuracy is independent of the number of predicted classes, it is not suitable for working with class-imbalanced data sets [56]. Thus, in order to leverage the $F1$

score metric, one may compute it in respect to every class the multi-class classifier makes predictions for and then take an average, for example arithmetic mean, over the scores for each class. Using arithmetic mean and n classes, *average* $F1$ *score* will be defined as:

$$\mathrm{F}_1^* = \frac{1}{\mathrm{n}} \sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{F}_1(\mathrm{i}) \tag{2.6}$$

2.5.3 Experiments and results

This section explains the experimental setup of the study on anomaly detection in driving behavior, including data acquisition for novel naturalistic driving patterns. All the implementations in this study were coded in Python, using Scikit learn [41]. To have a comprehensive measure of our model's performance throughout the whole data set, a shuffled train-test split function from scikit learn [41] is used on the data set. Furthermore, the details of the applied approaches and their results are in the following subsections presented.

Data acquisition

One of the main contributions of this study was proposing a complete ML framework that includes supplying a novel data set for anomaly detection in driving patterns based on multimodal sensor data in cars. In order to provide a labeled data set containing dangerous driving events for anomaly detection in driving patterns based on multimodal in-car sensor data, some definitions are needed. According to [57], a driving event usually refers to maneuvers occurring during the driving task, such as acceleration, deceleration, turning, and lane change. By extension, a dangerous driving event would then be such a maneuver if it is associated with an elevated risk of causing harm to the driver, other traffic participants, the vehicle, or other property. A driving pattern has previously been defined as the speed profile along with all analysis results that may be derived from it [57]. This study extends utilized sensor modalities containing other recorded signals, indicating speed, brake pressure, steering-angle, and derivatives, including frontal acceleration.

The data collection is conducted by the Chair of Operating Systems and Distributed Systems at the University of Siegen in collaboration with the company INVERS [2] as part of the LEICAR Project (section 1.1.2). Recorded data is based on four abnormal hazardous driving behaviors recreated by seven drivers with basic up to professional driving skills on a training ground in Olpe, North Rhine-Westphalia, Germany. Table 2.5 lists the vehicle models and types used in this experiment.

The hardware used for data collection consists of two Raspberry Pis with CAN-Bus adapter shields, which are connected to the cars' CAN-Bus ports. A GoPro camera is installed on the dashboard of the cars facing the windscreens, and an Empathica E4 wristband [58] are used by the drivers to mark the start and end of events (driving). The Empatica captures the driver's heart rate, skin conductance, and accelerometer. Only collected signal values from the cars are used in our study.

The Empatica wristband was connected to the Raspberry Pi via Bluetooth. Using the Empatica button, it was possible to split the track of the recording data when the drivers or the scenarios changed. In this way, the only contact with the drivers was via the Empatica wristband, and the drivers were not interrupted or distracted during the data acquisition.

Table 2.5: Cars used during data acquisition.

Model	**Transmission**	**Propulsion**
Audi Q3	automatic	combustion
Ford Fiesta	manual	combustion
Opel Ampera E	automatic	electric
Toyota Auris	manual	combustion

To replicate abnormal hazardous driving patterns, four different abnormal driving scenarios (events) were conducted and recorded at the training ground, guided and supervised by experienced driving trainers from the training lot (see Table 2.6).

Table 2.6: Conducted driving events with corresponding labels.

Label	**Event**
0	N/A (Normal driving)
1	Hard braking
2	Hard braking and steering
3	Aquaplaning
4	Slalom

These four abnormal hazardous driving scenarios have been designed to be reproducible. Their explanations and associated labels 2.6 can be found as follows:

- **Normal driving:** In this experiment, a normal driving scenario is considered the baseline for detecting anomalies in driving behavior. The normal driving events refer to smooth steering, braking, shifting, accelerating, and decelerating the vehicle at zero to 80 km/h in the practice area.
- **Emergency brake:** The braking scenario is performed by accelerating to 50-80 km/h and suddenly pulling the emergency brake.
- **Emergency brake plus steering:** Emergency braking plus steering is performed similarly to the previous scenario, in addition to forced steering after a full braking maneuver. These two scenarios are modeled on a driver's behavior

reacting to sudden environmental changes, e.g., an obstruction looming before him.

- **Aquaplaning:** The aquaplaning scenario was recorded on a slippery track with water sprinklers and a kick plate. Electronic Stability Control (ESP) was disabled on all cars to record this scenario, enabling us to capture the driver's reaction to the sudden loss of control over their vehicle.
- **Slalom:** The slalom scenario aimed to simulate aggressive driving behavior in a way that the drivers steered around a set of pylons at increasing speeds, trying to approach the limits of their ability to control the vehicle.

Data labeling

Assigning labels to the captured data is the most time-consuming and resource-consuming task in the data acquisition process. The accuracy of the label assignment can strongly influence the data quality and, thus, the efficiency of the supervised ML models.

In order to provide a suitable application for a proper labeling procedure, a labeling tool called supervisor was proposed by Hasse in [59]. The video recordings for each driving track were used to identify the time intervals that are indicators of a hazardous event, along with a graphical representation of the data for each signal recorded with the Supervisor application. Using these modalities and depending on the scenario, different cues were used to determine the start and end time index of the event intervals. In addition to the manual notes taken during the events and the synchronized video recordings, the parameters listed in Table 2.7 were considered in the label assignment.

Table 2.7: Labeling start and end cues.

Label	Event	Start	End
1	Hard braking	deceleration start	velocity = 0
2	Hard braking and steering	deceleration start	velocity = 0
3	Aquaplaning	kick plate activation	velocity = 0
4	Slalom	start/ end of periodical steering-angle pattern	

Data pre-processing and segmentation

The recorded data containing speed, acceleration, steering-angle, and braking pressure of the dangerous driving events are pre-processed and segmented according to the following steps:

- **Data cleaning:** Data cleaning is the procedure of adding missing data and correcting, repairing, or removing incorrect or irrelevant data from a data

set. Due to the connectivity issues during the recording, some of the recorded CAN-Bus traces contained no data. Others contained several significant gaps for unknown reasons. With this procedure, CAN-Bus traces falling into either of those two categories were discarded.

- **Data synchronization and aggregation:** Within this step, the video records corresponding to each CAN-Bus trace were used along with a graph representation of the data for each signal, and timestamps were synchronized. The speed, brake, and steering-angle signals were semi-automatically extracted from the CAN-Bus traces developed as part of the LEICAR project (see 1.1.2), resulting in a two-column text file for each signal - with the first column containing the time indices and the second column containing the value for each time index. The acceleration signal was then computed as the first derivative of the speed signal.

- **Data resampling:** Though the original signals were available at a sample rate of around 50 Hz on average, the choice was made to downsample them to 20 Hz in order to reduce the amount of data to process in further steps. In order to achieve this, a simple linear interpolation approach was used. Some of the signals had one more data point than most of the other signals in a given trace, presumably because the signals were not sampled in phase at recording time. Those excess data points that only became apparent after resampling were discarded.

- **Data normalization:** At this stage, Min-Max normalization was utilized, adjusting the signal values to a range from 0 to 1. Although this normalization choice has reduced the effectiveness of certain handcrafted features, the classifier is still capable of distinguishing between classes with various features, owing to the substantial inter-class variabilities observed in hazardous driving events.

- **Data segmentation:** As in [60], a sliding time window was chosen for the segmentation. Therefore, suitable values had to be chosen for the segment length and the step size of the sliding window (see Figure 2.2). For the segment length, 3s was initially chosen as an estimate for a period containing enough information for a system to make a classification decision. In later steps, segment lengths of 1s, 2s, and 4s were used to validate this choice. Of these choices, a segment length of 2s provided the most accurate classification results. The $\sigma = 1$ sample was chosen for the sliding window stride. Figure 2.11 presents one of the resulting segments of a driving sample recorded in this study.

As the data was labeled before resampling, labels had to be resampled as well in order to align them with the rest of the data, which resulted in non-integer label values for some time indices. A label mapping using thresholding was applied to adjust the suitable labels.
Let l_{max} be the numerical value of this label. Let $(l_t)_{t\in\mathbb{N}_0}$ denote the sequence of time index-based, potentially non-integer label values $l_t \in [0, l_{\max}] \subset \mathbb{R}$ for one file

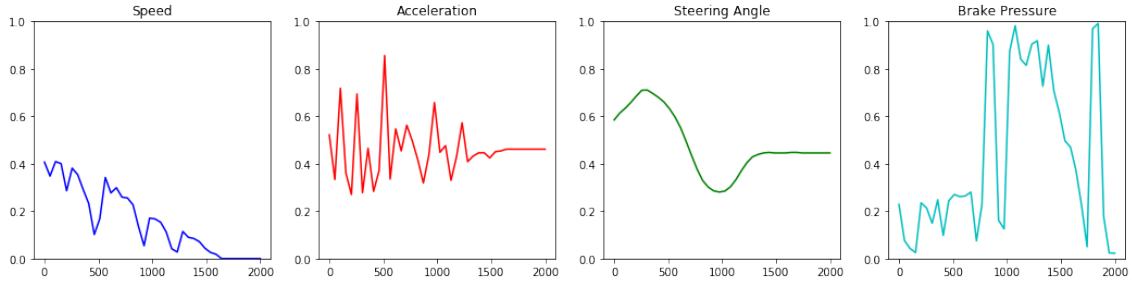


Figure 2.11: A segment of the aquaplaning scenario (label 3) after normalization.

that contains data from only one scenario with label l_{max}. The label l_s for a segment $s = [t_i, t_j], i \leq j$ is then determined by:

$$l_s = \begin{cases} l_{\max} & , \text{ if } \forall l_t \in s : l_t > \frac{1}{2} l_{\max} \\ 0 & , \text{ otherwise} \end{cases} \tag{2.7}$$

Feature extraction for anomaly detection in lab data

The HC features in this study are from most commonly used HC features on the time-series data as listed in Table 2.8. These features are computed individually on each sensor channel. The feature vector of each sample is then a concatenated vector of all extracted features of all sensor channels.

Table 2.8: List of the handcrafted features used on the Lab data. Each feature is computed on each sensor channel independently.

Handcrafted Features		
Maximum	Percentile 50	First-order mean
Minimum	Percentile 80	Norm of the first-order mean
Average	Interquartile	Second-order mean
Standard-deviation	Skewness	Norm of the second-order mean
Zero-crossing	Kurtosis	Spectral energy
Percentile 20	Auto-correlation	Spectral entropy

Classification

To classify the above-mentioned extracted feature, SVM is applied. SVM is advantageous because it can handle non-linearly separable data by transforming it into a higher dimensional space, where a hyperplane can separate the data. SVM also has a regularisation parameter C that helps prevent data overfitting, making it a reliable algorithm for classification tasks. Additionally, SVM is effective even when the number of features is larger than the number of samples, which is common in many real-world applications.

The chosen kernel of the SVM in this experiment is the linear kernel, a variant of SVM that uses a linear function to find the hyperplane that separates the data points. The linear kernel is the simplest and works by computing the dot product between two vectors in the input space. The decision boundary of the SVM with a linear kernel is a straight line in two dimensions, a plane in three dimensions, and a hyperplane in higher dimensions. The linear kernel is efficient and fast to compute, making it suitable for large datasets. However, the linear kernel can only separate linearly separable data, which means that it may not be suitable for some complex data sets. Nonetheless, SVM with a linear kernel is still a popular choice for many classification tasks because of its simplicity and effectiveness.

In this experiment, the linear kernel SVM were trained with the regularization parameter $C = 2^k$ for $k \in [1, 5]$. The C parameter in SVM controls the trade-off between maximizing the margin and minimizing the classification error. A smaller value of C will result in a wider margin, which can lead to more generalization but may also result in more misclassifications. Contrarily, a larger value of C will result in a smaller margin, which can lead to overfitting but may also result in fewer misclassifications.

Results and analysis

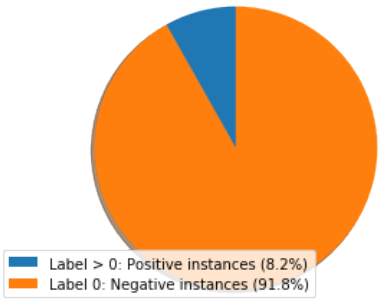
The results of the aforementioned feature extraction approach with the linear SVM classifier using different C parameters are provided in Table 2.9.

Table 2.9: SVM approach classification performance metrics in percent for different values of the C parameter. F1-score are shown for each label, followed by average and weighted F1-score and accuracy.

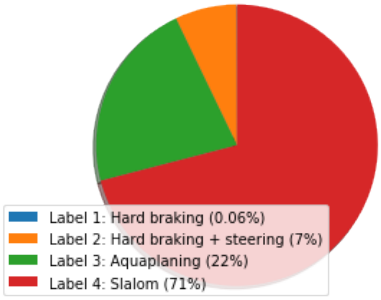
Measure	**C=2**	**C=4**	**C=8**	**C=16**	**C=32**
$F_1(0)$	97.65	97.65	97.62	97.60	97.21
$F_1(1)$	50.00	50.00	80.00	80.00	80.00
$F_1(2)$	78.66	79.55	79.17	80.07	70.13
$F_1(3)$	65.29	65.62	65.17	66.94	64.40
$F_1(4)$	68.98	69.07	67.67	66.91	71.00
Average F_1	72.12	72.38	77.93	**78.31**	76.55
Weighted F_1	95.23	95.25	95.13	95.10	94.88
Accuracy	95.58	95.60	95.54	95.50	94.84

Table 2.9 reveals various observations. First, as noted before, the accuracy measure does not seem helpful within the context of the class-imbalanced data set, producing a misleadingly high value of 95.58%. Thus, the average $F1$ score was used to identify the best-performing run for each implementation. Secondly, SVM's Average $F1$-score indicates the best performance of 78.31% with the regularization parameter $C = 16$. Moreover, the performance gap between $F1$ scores of different classes, most obviously the $F1$ score 97.60 % of the class containing normal driving, $F1(0)$, events compared to the one by class aquaplaning, $F1(3)$, with 66.94% needs a glimpse of the quantity of the classes. Figure 2.12, shows two charts showing the class proportions

based on negative to positive driving events and the distribution of various scenarios in positive events, respectively.



Class balance: Positive versus negative instances.



Distribution of positive class instances.

Figure 2.12: Class distribution of the recorded data set, measured after pre-processing and segmentation.

As previously noted, although we explicitly attempted to capture dangerous driving situations, the class balance is still dramatically tilted toward negative instances. This imbalance can be observed in Figure 2.12(a), which explains the high $F1(0)$(the normal driving event). Delving deeper into the class distribution among the positive instances, we can observe a substantial imbalance even within the positive classes (see Figure 2.12(b)), which explains the performance differences of the SVM classifier of each class.

The quality of the labeled data is the second important fact to be noted before concluding the presented framework's performance. Creating an actual data set with high-quality labels has been proven to be the most challenging part of a PRC framework. In this case, several challenges were encountered, which clearly impacted the model's performance. Since the windshield camera failed several times during the recording process due to overheating problems, labeling was made considerably more difficult in these cases and may thus have suffered in terms of accuracy.

Another obstacle that had to be overcome was that the recorded CAN-Bus trace files needed to be marked with usable file system timestamps. This made matching the video recordings with CAN-Bus data considerably more difficult. In the cases where video recordings were needed to identify an event's start, data and video files had to be matched based on their duration manually. In the case of aquaplaning, extensive use of the video material recorded using the windshield camera was required. This was primarily because the kick plate activation did not produce an easily distinguishable pattern in any signals.

In summary, it can be said that, given the circumstances, the class distribution of the data set and the accuracy of the labels turned out to be quite adequate

and rendered the data set suitable for use in developing and evaluating dangerous driving detection systems.

2.5.4 Conclusion

This subsection introduces a labeled data set of naturalistic hazardous driving behavior for detecting anomalous driving patterns using primary in-car sensor data. Five driving patterns, including normal driving, driving with the emergency brake, emergency brake plus severe steering, aquaplaning, and slalom, by seven drivers with different driving skills driving four car models are performed in a training lot in Germany. The data are labeled via a labeling tool (supervisor [59]). The proposed PRC framework (see 2.3) is used to classify data in a multi-class classification task. The results presented in subsection 2.5.3 reveal a notable 78.31% average $F1$ score with HC features classified by SVM. Despite various challenges in the data acquisition procedure or the labeling task, in addition to the huge obstacle of dealing with imbalanced data, this work greatly contributes to exploring and benchmarking different types of dangerous driving event detection systems.

Last but not least, there are a few opportunities to improve further this study investigation in the domain of dangerous driving behavior detection using ML pipelines. To produce a data set to be used in a real-world dangerous driving event detection application, the data acquisition approach needs to be improved threefold.
Firstly, events with a comparatively short duration, such as hard braking, may have to be recorded disproportionately often to even out the class imbalance, as depicted in Figure 15b. Secondly, additional care must be taken to ensure the recording equipment stays operational during data acquisition sessions. More readily available video material as well as an easier means to match that video material with time indexes found in the CAN data would have significantly contributed towards creating more accurate labels.

For the PRC framework, the feature extraction stage is a good starting point for optimization. In particular, studies such as [60] have found that using an SVM in combination with a Convolutional Neural Network (CNN) Long Short-Term Memory (LSTM) hybrid or a fusion of both of these approaches for automated feature extraction produces improved time-series data classification performance. In addition, employing a kernel other than the linear kernel used for the SVC in this study, for instance, an RBF kernel, might produce better results.

2.6 Scientific discussion

This chapter investigates anomaly detection in driving patterns based on in-car sensor data. The chapter examines anomaly detection in driving patterns via time-series data in-car sensors. The first section of this chapter proposes a PRC framework to apply ML approaches as a baseline for anomaly detection in time-series in-car sensor

data. By using this framework, a comparison of the supervised and unsupervised anomaly detection on the in-car sensor data is performed. Furthermore, in this chapter's second section, the number of in-car sensor modalities is expanded to suffice to utilize a deep learning algorithm. Since the acquired data are unlabeled, an unsupervised deep learning anomaly detection technique is applied.
Traditional handcrafted feature extraction and SVM and kNN classifiers are used to detect abnormal driving patterns on the speed time-series. Supervised kNN classifier achieves a notable 98.03% F1-score while kNN-unsupervised and SVM gain 87.85% and 84.84% F1-score, respectively.
In addition to the abovementioned approaches, cluster-based ML approaches are applied too. K-mean and Gaussian mixture models are implemented to detect anomalous driving patterns in the speed sensor data. GMM surpasses the K-mean clustering with a 94.17% F1-score. It is clear that the high classification score of the supervised kNN is due to the learning with labeled data.
The notable results of the performed experiment show an apparent success of the proposed PRC framework. Although the acquired data rarely contain severe abnormal driving patterns, the results reveal an evident success in finding anomalous driving patterns detecting the least frequent driving events, such as driving on the speed bumpers or standing behind the traffic light as abnormal driving patterns compared to normal daily driving.
Based on the notable success of the proposed methodology for detecting abnormal driving patterns using in-car sensor data, further improvement of this research investigation is only possible with sufficient labeled data containing severe abnormal driving patterns. Therefore, the final section of this chapter provides a novel driving data benchmark acquired based on naturalistic driving events performed on a training lot. The proposed PRC framework is then applied to the acquired data using HCF extraction with an SVM classifier. Five driving events are classified via a multi-class classifier, including normal driving, hard braking, steering, aquaplaning, and slalom. The reason for not using a deep learning approach in this experiment is the insufficient data for training and testing purposes. The results reveal a notable average F1-score of 78.31%. This result is prospering since this performance is based on a multi-class classification task.
The presented chapter's overall observation of these experiments shows that it is possible to detect abnormal driving patterns only based on primary in-car sensor data. Using ML tools on a sufficient amount of data that the anomalies are well defined is the primary key to conducting a successful classification task of detecting abnormal driving patterns with in-car sensor data.

2.7 Summary

Chapter 2 presents a novel study, filling the gap in the literature on driving behavior anomaly detection based on time-series data in cars. The lack of labeled data and the scarcity and complexity of anomalous driving patterns are the primary reasons for the research gap in the field of driving anomaly detection based on in-car sensor data. Identifying abnormal driving patterns is crucial for recognizing accidents

and near accidents. When examining all anomalous driving patterns within a data set, accidents represent the least frequently occurring events in contrast to normal driving events. Therefore, the ability to distinguish abnormal events from normal ones is a big step towards recognizing certain abnormal events as accidents.

This chapter delved into anomaly detection in driving data in three main sub-sections. The first section introduced a baseline framework for PRC anomaly detection in driving patterns using speed data from CAN-Bus inside cars. The state-of-the-art ML algorithms were applied and analyzed, including supervised (KNN and SVM) and unsupervised (K-mean and GMM) approaches. Based on the results and concerning the anomaly detection quality, the supervised KNN achieved the best performance with 98.03% $F1$-score. In the unsupervised case, the GMM algorithm obtains 94.17%, the second-best $F1$-score.

In the second section of chapter 2, the exploration of anomaly detection in driving patterns continued by leveraging unlabeled CAN-Bus signal values from twenty-five driving trips across different cars. This extended investigation, conducted within the proposed PRC framework, not only pushed the boundaries of unsupervised anomaly detection in in-vehicle sensor data but also paved the way for potential future research endeavors. The provided framework significantly impacts solving the problem of anomaly detection in driving patterns. Based on the state of the collected data (enough data for ANN models but unobtainable labels), different variations of AEs, one of the most popular unsupervised ANNs, are tested and analyzed. According to the observed illustrations of the detected anomalies in speed data, all AE variations perform similarly stable. The choice of the threshold impacts the final detection in all three variations of AE. The performance of AEs on the multi-modal CAN-Bus signals, including speed and brake, could have been more outstanding than a single modality. However, LSTM-AE outperforms other AEs in detecting anomalous driving patterns based on speed and brake signals.

In the concluding part of chapter 2, a benchmarking data set is introduced to detect anomalous driving patterns using primary in-car sensor data. The data set involves five distinct driving patterns performed by seven drivers across four car models, encompassing everyday driving, emergency braking, emergency braking with severe steering, aquaplaning, and slalom, conducted in a training lot in Germany. Supervised labeling through a labeling tool was employed, and the proposed PRC framework (see Figure 2.3) was utilized for multi-class classification. The achieved results in section 2.5.3 demonstrate a noteworthy 78.31% average F1-score with HC features classified by SVM, contributing significantly to the exploration and benchmarking of diverse dangerous driving event detection systems, despite challenges in data acquisition, labeling, and imbalanced data.

Overall the results of this study highlighted the following main points:

- The anomaly detection in driving behaviors based on in-car sensor data requires explicit knowledge of the data and the answers to the following questions. 1. What is an anomaly concerning the nature of the data set? 2. What

are the characteristics of the anomalous events we are seeking? 3. Can we define the anomalies directly, or should we look at how the ML models classify them and base our definition on them?

- In the case of unlabeled data, despite the unpredictability of the data, the unknown shape of anomalies, and their ever-changing range in the collected data set of driving trips, AEs show coherent and promising performance in finding anomalous driving patterns based on CAN-Bus signals.
- In case of labeled data, section 2.5 reveals a notable 78.31% average F1-score with HC features classified by SVM.
- Finally, the data's quality and quantity are prerequisites for a successful anomaly detection model. Therefore, section 2.5 introduced a benchmarking naturalistic data set of hazardous driving behaviors, significantly contributing to anomaly detection in the driving behavior analysis domain.

Chapter 3

Automated car accident detection based on multimodal sensor data

The demand for transportation in developing countries is on the rise, leading to a significant increase in road traffic injuries and fatalities, particularly in low- and middle-income nations [61]. According to the World Health Organization (WHO) [62], the substantial toll of injuries and fatalities resulting from traffic accidents underscores a global road safety crisis, with such accidents ranking among the leading causes of death for individuals aged 5 to 45. Despite ongoing advancements in accident prevention and detection systems within the automobile industry, there remains a pressing need for efficient automated accident detection to help save lives. Furthermore, accident detection has significant implications for environmental and safety applications, as well as the expanding field of fleet management. It plays a crucial role in reducing the risks associated with both vehicles and drivers, enhancing service quality, and cutting operational expenses. While previous literature has proposed solutions for automated accident detection primarily relying on traffic data or external sensors, there are challenges. Accessing traffic data can be complex, and the setup and reliability of external sensors can be problematic depending on their deployment. Moreover, the scarcity of accident detection data in the past has limited the variety of approaches, with machine learning (ML) being relatively unexplored. Consequently, this chapter introduces an ML framework for automated car accident detection, leveraging multimodal in-car sensors and state-of-the-art feature extraction methods.
To summarize, the contributions of this chapter are as follows:

- The content discussed in this chapter marks a groundbreaking exploration, being the inaugural investigation into ML-driven accident detection using fundamental in-car network data. This study stands as a distinctive and pioneering research endeavor, as it focuses on identifying actual driving accidents using the most readily available and cost-effective data sources within vehicles.

- Within this chapter, an in-depth ML framework is unveiled, built upon the PRC framework depicted in Figure 3.1, designed to execute accident detection by leveraging fundamental in-car network data. Furthermore, this framework is employed to conduct a comparative analysis of cutting-edge ML feature ex-

traction techniques, specifically tailored for in-car sensor data used in accident detection. The framework is applied to the SHRP2 NDS crash data set, which encompasses data from gas-pedal position, speed, steering-angle, and acceleration sensors. This comprehensive approach yields promising results in the realm of automated accident detection using naturalistic data.

This chapter includes content previously published by Hozhabr Pour et al. in [11]. The chapter is structured as follows: Section 3.1 addresses the problem statement and underscores the need for an ML-based framework for automatic accident detection using in-car sensors. Section 3.2 reviews related work in the field. Section 3.3 describes the proposed ML framework, illustrated in Figure 3.1, and briefly explains the materials and techniques employed. Section 3.4 discusses the implementation details of all applied algorithms and explores the results. In section 3.5, we present the scientific interpretation of this study. Section 3.6 serves as the conclusion and summary, addressing accomplishments, study limitations, and future work.

3.1 Problem statement

The research focus on identifying accident patterns is paramount within the realm of driving analysis. As per the global status report on road safety conducted by the WHO [62], the number of traffic-related fatalities continues to steadily increase. Despite notable advancements in road safety through initiatives like Driver Assistance, Safety Awareness Services, and Automatic Crash Notification (ACN) systems, the significance of accident detection and prevention in driving studies remains undiminished. Consequently, accident detection studies have garnered the interest of insurance and fleet management companies [63, 64]. Furthermore, the impact of accident analysis extends to various domains, including environmental, road safety, and commercial applications such as insurance and loan qualifications, where its contributions hold significant potential [65].

An important contribution of the accident detection studies is the post-crash applications concerning immediate dispatch of the emergency and roadside assistance services [66, 67, 68, 69]. However, these studies do not analyze accident patterns but rather imply factors such as airbag deployment to detect an accident. A recent application of accident detection, which studies accident patterns, is automated boxes installed in cars that minimize the service and overhead inspection cost of fleet operators by detecting car damages due to minuscule accidents, bumps, accelerations and braking manoeuvres [70].
In the domain of accident detection, studies employing machine learning (ML) are relatively scarce, as noted in a review by Meiring [65]. Several factors contribute to this scarcity. Firstly, in cases involving accident detection, providing positively labeled data is exceedingly challenging. Furthermore, creating a large real-world database containing accident events is not only costly and time-consuming but is also hampered by the competitive nature of the automobile industry and data privacy concerns.

To the best of my knowledge, the most substantial database for real-world accident data, including vehicle time-series, is the one provided by the second Strategic Highway Research Program (SHRP2) Naturalistic Driving Study (NDS) [71]. Data scarcity in the field of accident detection is a significant impediment to developing an automated accident detection system trained with ML, as such systems necessitate large, comprehensive, unbiased, and high-quality data sets. Moreover, waiting for new data to be generated is often required.
The complex nature of accident events itself presents substantial challenges. In cases like these, manually labeled training data is essential, where precise conditions characterizing the definition of an accident must be predefined. According to the Virginia Tech Transportation Institute (VTTI) [71], an accident is defined as "any contact that the subject vehicle has with an object, either moving or fixed, at any speed in which kinetic energy is measurably transferred or dissipated. This also includes non-premeditated departures from the road, as well as instances where the subject vehicle strikes another vehicle, a roadside barrier, a pedestrian, cyclist, animal, or object on or off the roadway."
Based on this definition, even minor incidents, such as minor tire contact with low or no risk (e.g., clipping a curb during a tight turn), can be considered accidents. Identifying such cases typically relies on incident reports or thorough inspections, posing a significant challenge for automated accident detection.
Accident detection is typically framed as a binary classification problem, where input data are used to train models representing accident and non-accident classes. ML-based accident detection studies can be categorized based on the type of data used to train their models.
Two broad categories emerge in the field of accident detection: one relies on traffic data, while the other leverages external sensors such as smartphones, acoustic sensors, or cameras [65]. However, the performance of prediction and detection systems in these categories is heavily constrained by factors such as sensor availability, budget, weather conditions, and traffic flow. The use of external sensors can also pose challenges regarding installation and reliability, depending on their deployment [72].
For the aforementioned reasons, internal car data have emerged as a promising third option for ML-based accident detection, free from the limitations mentioned earlier. In this context, car in this work refers to a passenger vehicle. Modern cars, manufactured from the mid-1990s onwards, come equipped with an array of sensors that provide reliable time-series data for fundamental driving attributes, including speed, steering-angle, and gas pedal position. This data holds significant potential for accident detection and is readily accessible by monitoring the in-car network.

For example, the on-board diagnostics (OBD) adapter represents the most accessible means to capture driving patterns through network protocols like CAN-Bus inside cars. Additional information on classifying and deciphering the meaning of the most vital signals transmitted within the in-car network can be found in the previous study in section 1.1.2.
Machine learning (ML) can be applied to in-car network data by following the established PRC framework, which is comprehensively described in 2.1.2 and comprises four key steps, as depicted in Figure 3.1.

1. Data Acquisition: In the first step, the selection of sensors is based on the nature of the classes being studied and their availability.

2. Data Pre-processing: The second step involves data pre-processing, including tasks such as sensor calibration, unit conversion, normalization, and segmentation. These operations are carried out to render the data suitable for further analysis.

3. Feature Extraction: The next step, feature extraction, is focused on deriving the most pertinent information from each data segment. This process results in an abstracted and informative representation of each data segment, enabling more effective analysis.

4. Classifier Training: In the final step, classifiers are trained to distinguish between different classes within the feature space. In this specific case study, the goal is to differentiate between accident and non-accident driving events.

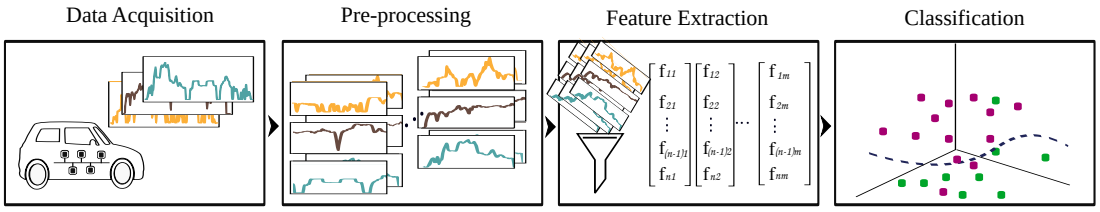


Figure 3.1: Machine learning framework for accident detection. First, time-series data is being acquired from in-car network signals. After pre-processing and segmentation, different feature extraction approaches are applied and compared. Finally, classifiers are trained and tested on the extracted features.

Past experience has demonstrated that each of these steps significantly influences accident detection performance [73]. However, it's worth noting that data acquisition and pre-processing are primarily dependent on the specific data set, while classification relies on established state-of-the-art approaches with proven reliability from prior research.
Consequently, it is in the step of feature extraction where the potential for improvement is most substantial and has been the subject of extensive investigation in previous ML studies. Therefore, conducting a study on applicable feature extraction methods holds considerable importance in providing an ML framework for accident detection.
Previous research on feature extraction primarily focuses on two types of techniques: feature engineering and feature learning.

- Feature Engineering: This is a conventional approach to feature extraction that relies on prior expert knowledge of the data. It involves crafting features

manually, specifically tailored to the problem at hand. An example of feature engineering is the handcrafted method, which computes simple statistical values as features from the input data or features designed by experts to address a particular problem [60, 74, 75]. However, feature engineering may not always be the most optimal approach, especially when expert knowledge is lacking, and there's no assurance that the selected features will be ideal for the task.

- Feature Learning: Feature learning is an automated process of feature extraction using artificial neural networks (ANNs), which are machine learning models known for their capability to extract highly efficient features. This is contingent upon conducting appropriate pre-processing and segmentation steps on the data [76]. However, working with ANNs can be intricate due to the need to find the right parameters and effectively train the model. Additionally, the features learned by ANNs are often challenging to interpret, which can pose a significant challenge in various applications.

Hence, providing a comprehensive study that explores the application of both feature engineering and feature learning approaches, and analyzes the optimal feature extraction methods despite their respective limitations, is of immense importance. Despite the popularity of ANNs in obtaining efficient features, finding the proper parameters and properly training the model can be complicated. Additionally, the features learned by ANNs are usually difficult to interpret, which can be a major obstacle to their use in many applications. Therefore, providing a comprehensive study on applying feature engineering and feature learning approaches and analyzing the optimal feature extraction approaches in spite of their respective drawbacks is of tremendous importance.

3.2 Related work

The growing demand for mobility has made driving behavior analysis applications a pivotal area of research. The outcomes of driving behavior analysis hold substantial significance for a range of sectors, including the automotive and intelligent transportation industry, automobile insurance, and government organizations overseeing infrastructure and public transportation. Numerous studies highlight the importance of driving behavior analysis in the context of traffic management, safety, and environmental concerns [77, 78, 79, 29]. Simultaneously, numerous other studies focus on the analysis of driver behavior itself [65, 80].

Due to the diverse array of research objectives, applications, contributions, and data sources, there is no specific study baseline or well-defined research categorization in the domain of driving behavior analysis, especially concerning accident detection.To comprehensively review the recent state-of-the-art in accident detection, it is essential to provide an overview of relevant works within the broader field of driving behavior analysis. While time-series feature extraction has been extensively explored [81, 82], it's noteworthy that the findings from these feature extraction studies do not always translate seamlessly from one application domain to another.

Consequently, this specific topic has received relatively limited attention in the context of accident detection. Nonetheless, the relevant work regarding time-series feature extraction used in driving behavior analysis is provided in section 2.2.

In this section, two primary categories of studies are presented. The first category offers an overview of the state-of-the-art in "driving behavior analysis" as a whole, while the second category is dedicated to the literature on "accident detection".

3.2.1 Driving behavior analysis

Numerous previous studies have conducted comprehensive surveys within the field of driving behavior analysis. One such study by Zinebi et al. [80] categorized driver behavior analysis into three primary sub-applications: accident prevention, driving style assessment, and driver intent prediction. Additionally, they identified three categories of methods commonly used to address these challenges: index systems, image processing, and statistical methods and machine learning.

Index systems, as one of these categories, are focused on defining specific metrics or indices to objectively quantify high-level concepts associated with driver behavior. An example of such an index is TTC (time to collision), introduced by Mori et al. [83], which measures the time a vehicle takes before colliding with an object in its environment. They used this index to calculate the environmental risk score at a given time. Notably, they found a correlation between driver attention and the environmental risk score, highlighting that expert drivers tend to exhibit higher levels of awareness compared to non-expert drivers.

Image processing represents another widely adopted method for driving behavior analysis, primarily due to the increased availability of vision sensors. An example of image processing in the context of driving analysis is the detection of driver drowsiness using computer vision techniques from an iPhone, coupled with GPS for tracking the car's position, as demonstrated in the work by Bergasa et al. [63].
The final category of methods in the reviewed literature, as presented by Zinebi et al. [80], encompasses statistical and machine learning (ML) approaches. For instance, Bachoo et al. [84] utilized multiple linear regression to investigate the impact of personality traits (e.g., anger, impulsivity) on reported instances of risky driving behavior using a cross-sectional questionnaire.

In another study, Jahangiri et al. [85] employed a random forest (RF) classifier to categorize driver behavior into two classes: violation and compliance at signalized intersections. They leveraged various features, including distance to the intersection, velocity, acceleration, time to the intersection, required deceleration parameter, and velocity-based handcrafted (HC) features obtained from radar, video cameras, and signal phase sniffers at intersections. Their models achieved high accuracies of 97.9% and 93.6% for SVM and RF, respectively, in predicting driving violations at signalized intersections.

Ohn et al. [86] adopted a support vector machine (SVM) to classify driver activities based on hand positions, using cameras placed within the vehicle. They employed hand motions for activity classification and prediction from naturalistic driving images. Image-based features were used to track hand motions and detect various hand patterns in different regions of the car, with the SVM achieving a normalized accuracy of over 80%.

All the ML-based driving behavior analysis studies reviewed in [80] are focused on the detection of various types of driving behaviors, activities, and driving styles. None of these studies specifically pertain to accident detection. In a separate comprehensive study conducted by Meiring et al. [65], the focus is on driver style analysis systems, their applications, and the artificial intelligence algorithms that underpin these applications. This review encompasses two major categories of driving research: driving style and studies assessing driver behavior. Within the papers reviewed in [65], various AI, ML, and statistical algorithms are applied to diverse driving research topics, including driver assistance, drowsiness detection, driver distraction detection, eco-driving, road and vehicle condition monitoring, fleet management, accident detection, and insurance applications.

The majority of the literature mentioned in [65] primarily centers around the detection of different driving styles, encompassing categories such as normal and safe driving, aggressive driving, inattentive driving, and drunk driving. Various data modalities are employed in these studies, including multiple sensors like car networks, smartphones, telematics, and video data. However, there is a notable scarcity of literature specifically addressing accident detection. One example of a study on accident detection is presented by Lee et al. [87]. In their work, they proposed a log-linear model to predict crashes based on crash precursors, specifically traffic flow conditions leading up to the crash. This predictive model was constructed using traffic flow data obtained from traffic loop detectors.

In their study, Bagdadi et al. [88] introduced a method for recognizing critical jerks using the naturalistic Virginia Tech Transportation Institute (VTTI) near-crash data. They achieved a detection rate of 86% for identifying safety-critical braking events during car driving. Critical jerks, as defined in their work [89], refer to sudden changes in acceleration magnitude. Safety-critical braking events are identified as events involving an abrupt braking response, leading to the creation of a critical jerk.

3.2.2 Accident detection

To facilitate a more straightforward comparison with prior research, the related works from the literature have been categorized into two distinct groups: rule-based and ML-based accident detection studies.

Rule-based accident detection

Rule-based approaches are essentially straightforward problem-solving techniques often relying on heuristic rules that draw from expert knowledge. These methods are tailored to address specific problems and are most effective when applied to particular data modalities and intended problems. In the realm of accident detection, several research studies have been conducted on rule-based systems, particularly those centered on traffic-monitoring data [90, 91, 92, 93].

Traditional traffic accident prediction typically relies on annual average traffic volume data. In contrast, real-time traffic accident detection employs monitoring devices such as induction loops, infrared detectors, and cameras. However, the practicality of these specific devices is constrained by the considerable installation and maintenance costs, as well as their limited coverage of road networks, which is typically confined to well-known congestion-prone areas like highways, tunnels, or bridges [72].

Conventional built-in automatic accident detection systems make use of impact sensors or car airbag sensors to identify accidents and GPS technology to pinpoint the accident location [94]. Sheu et al. [95] introduced a novel approach for real-time detection and characterization of freeway incidents. In this system, incident symptoms are extracted from raw traffic data, encompassing segment-wide inter-lane and intra-lane traffic dynamics, lane-changing fractions, and queue lengths. These symptoms are identified through signal processing techniques, including extended Kalman filtering and the modified sequential probability ratio test (MSPRT).

Apart from traffic data, smartphones are one of the most commonly utilized sensors in rule-based accident detection research. For instance, Zaldivar et al. [66] introduced an application that automatically alerts emergency services about an accident through SMS, relying on the vehicle diagnostics interface (OBD-II). In this system, accidents are detected through airbag triggers, which are activated when a force overload is experienced during a frontal collision.

A slightly different approach, as proposed in [96], measures the change in tilt angle using an accelerometer sensor and monitors speed using GPS data to detect the moment of collision and trigger an alert when an accident is detected. Another method, presented in [97], focuses on using the smartphone's accelerometer to monitor vehicle speed and report an accident when it falls below a certain threshold. However, it is worth noting that smartphones may not always be reliable for accident detection. The main issue with these systems is that smartphones can tilt or fall inside the vehicle at any time without an actual accident occurring. As a result, the probability of a false positive increases, and false alarms may be reported. Past literature in ubiquitous computing has also revealed significant differences between smartphone brands, indicating that a machine learning model trained on one brand of smartphone may experience a substantial degradation in performance when used with a smartphone from a different brand, even if the latter contains the same sensors [98].

ML-based accident detection

Machine learning techniques exhibit notable variations in their algorithmic approach, depending on the type of input data they are applied to. Given the substantial data requirements of machine learning and the limited availability of labeled car accident data sets, only a small number of relevant studies are available. Among these studies, three primary sensor modalities are commonly combined with machine learning in the literature: traffic data, sensor data, and internal car signals.

Studies proposing accident detection based on machine learning approaches primarily rely on traffic monitoring data. For example, Ozbayoglu et al. [99] used traffic-flow data from the city of Istanbul in year 2015, obtained from real-time monitoring system (RTMS) sensors. They manually engineered specific features related to lane velocity, occupancy, and capacity usage for each sensor. These extracted features were then used as input for various machine learning models, including nearest neighbor, regression trees, and feed-forward neural networks, to predict the likelihood of an accident. It's important to note that while the overall accuracy of their models mostly exceeded 99%, a significant number of false alarms were generated.

Alvi et al. [73] conducted a comprehensive literature review related to Internet of Things (IoT)-based accident detection, prevention, and reporting systems. In this review, various applications of IoT were introduced and referenced, and accident detection papers were categorized into two groups: (1) conventional and (2) machine learning/artificial intelligence-based accident detection techniques.
For the second category of the reviewed accident detection papers in [73], three key machine learning/artificial intelligence-based approaches were discussed: (a) fuzzy logic, (b) Support Vector Machine (SVM), and (c) Artificial Neural Network (ANN). Pan et al. [100] utilized vehicle speed, acceleration, and lane changing factor data from a microscopic traffic simulator to classify incidents vs. non-incidents using an SVM. Their proposed methodology achieved an SVM accuracy of almost 100% based on speed data. It's important to note that their work assumes that each vehicle collects its own traffic data and transmits it through an On-Board Unit (OBU). Subsequently, the traffic data is gathered via Roadside Units (RSUs) and uploaded to a central service for processing. Their simulated scenarios only considered accidents on three-lane urban roads at traffic lights.

Harlow et al. [90] introduced a system that involves a method for processing and recognizing accidents based on recorded vehicle acoustic signals, particularly at intersections and construction sites. They created a database of various vehicle sounds, including car braking sounds, construction sounds, and traffic sounds. The feature vector was obtained by computing the mel-frequency cepstral coefficients, which were used as input for a neural network to classify events as either crashes or non-crashes. Their study reported classification testing results with an impressive 99% accuracy. However, it's essential to note that this system is designed for specific locations, such as intersections and construction sites.

Ghosh et al. [101] introduced an accident detection approach based on a CNN that utilizes video footage from CCTV cameras installed on highways. Their model, employing a combination of CCTV cameras and a Raspberry Pi 3, uses a pre-trained CNN model trained on 10,000 accident frames and 10,000 non-accident frames. This approach achieves an accuracy of approximately 95% in accident detection. However, it's important to note that this model is limited to accidents within the camera coverage area, potentially overlooking accidents occurring outside this scope. Additionally, the cost of monitoring all roads and highways with Pi cameras and Raspberry Pi can be significant. Privacy and security concerns also need to be addressed in such a system. Weather conditions can further impact the visibility of the cameras, affecting their reliability. The mentioned works highlight approaches that come with inherent limitations, including privacy and security concerns, high costs for large-scale networks, various manufacturers, industrial challenges, power consumption, architectural complexities, heterogeneity, mobility issues, and interoperability problems.

A solution similar to the one presented in this study, which utilizes internal car data to address the drawbacks of other monitoring devices, was proposed by Osman et al. [102]. This research introduces a machine learning model designed to predict collisions using vehicle kinematic data from the SHRP2 NDS data set, including parameters such as speed, longitudinal acceleration, lateral acceleration, yaw rate, and pedal position. The hypothesis behind this approach is that vehicles exhibit micro-level turbulence in their kinematic patterns in the period leading up to a crash, known as the turbulence horizon.
In their study, Osman et al. used the standard deviation of vehicle kinematic parameters during the period starting at the beginning of the turbulence horizon and ending at the beginning of the prediction horizon as input features. These features were employed to classify near-crash data and normal driving data, and they trained and compared several classification algorithms. While their model achieved an impressive 99% F1-score, it's worth noting that the feature extraction approach in this study relied on label re-computation. This method may not ensure that the extracted features are generic enough to be successfully applied to data sources other than the SHRP2 NDS data set. This study differs from the one by Osman et al. in two significant ways. First, the primary focus here is on accident detection rather than accident prevention, and, as a result, it does not include near-crash scenarios in the data set. Second, a noteworthy aspect of this work is the exploration of feature-learning approaches based on deep neural networks. These networks have the capacity to automatically learn features without requiring any prior knowledge or manual feature engineering.

3.3 Materials and methods

In this section, the proposed ML framework (Figure 3.1) will be introduced, providing a detailed explanation of its application for car accident detection using internal

car data. Since the most substantial improvements in classification performance are typically achieved during the feature extraction stage, the framework will be utilized to analyze and compare the respective performances of state-of-the-art feature extraction techniques. To facilitate this analysis, the acquisition, pre-processing, and classification steps will be maintained as outlined in the following subsections.

3.3.1 Data acquisition

Data acquisition involves defining an experimental protocol to configure sensors, annotating data with labels, sampling relevant sensor signals, converting samples into digital numeric values, and acquiring and merging data from appropriate sources.
Access to a suitable data set and data quality are essential prerequisites for the successful implementation of ML-based studies. Since this study centers on accident detection using real data, obtaining a suitable labeled database was a necessary initial step. In the conducted experiments, the decision was made to utilize the SHRP2 NDS database from the Virginia Tech Transportation Institute (VTTI) [71]. In the following section, a brief description of the SHRP2 data set will be provided.

SHRP2 data set

The SHRP2 research project focused on naturalistic driving behavior and involved the monitoring of approximately $3,400$ apprentice drivers using over 277 unique car makes/models. This study spanned six locations across the United States. The participants' vehicles were equipped with a comprehensive data acquisition system (DAS) that included various sensors and data collection tools such as a forward radar, four video cameras, a front-facing wide-angle camera, accelerometers, vehicle network information, a geographic positioning system, on-board computer lane tracking, various computer vision algorithms, and additional data storage capabilities.
An accident data set was compiled from a vast number of trip log files, totaling $5,512,900$ driving traces, extracted from the SHRP2 naturalistic driving study (NDS). To identify accidents, a team of data analysts and data quality coordinators manually validated and analyzed the log files to annotate the data. An accident was defined as any situation where a vehicle made contact with an object, whether moving or stationary, at any speed, resulting in the measurable transfer or dissipation of kinetic energy. This definition also encompassed non-premeditated departures from the road, where at least one tire left the paved or intended road surface.
The acquired data set includes 546 synchronized driving traces that contain accidents, and it contains various time-series data channels as shown in Table 3.1.

3.3.2 Data pre-processing and segmentation

Data pre-processing encompasses a series of operations aimed at rectifying flaws in the data arising from issues like data transmission errors or sensor failures. These operations include eliminating duplicates, addressing data irregularities, normalizing the data for comparison, and handling missing data values, a common issue in data sets. Ensuring the ML model receives consistent data is crucial for improving result

Table 3.1: SHRP2 data set sensor channels.

Variable Name	**Unit**	**Description**
Time stamp	millisecond	Time since beginning of trip, in milliseconds
Gas pedal position	none	Position of the accelerator pedal collected from the vehicle network and normalized using manufacturer specs
Speed network	km/h	Vehicle speed indicated on speedometer collected from network
Steering wheel position	degree	Angular position and direction of the steering wheel from neutral position

accuracy. Raw data records are typically lengthy and may lack uniform information, necessitating the division of data into shorter segments.
Segmentation is the process of breaking down signal values, especially time-series data, into distinct time intervals known as windows. The segmentation and pre-processing steps applied to the SHRP2 data set in this study are illustrated in Figure 3.2.

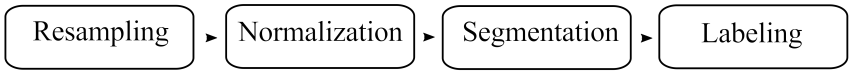


Figure 3.2: Pre-processing procedure applied on SHRP2 data set for accident detection.

To synchronize the sensor channels in the SHRP2 data set, the data was resampled from different sensors at a frequency of 40 Hz. Subsequently, min-max scaling normalization was applied to bring the sensor values into the same range. For segmentation, a traditional sliding time window approach was employed. The labeling process utilized information from event descriptions provided with the SHRP2 accident data set, which includes event start and end timestamps. Based on this information, the segment length (T) was set to 100, which corresponds to approximately 2.5 seconds. This duration was chosen to match the maximum event duration described in the SHRP2 data set. An overlapping factor of 50% was also applied to segment the time-series data.
The labeling process adds a label of zero or one to each time window depending on whether the frame at a specific timestamp is part of the event. Windows containing more than half of the duration of the frame dedicated to an accident event are labeled as one, and the rest of the windows are labeled as zero. After the aforementioned steps, the data consists of $34,339$ time windows, sized with dimensions $T \times S$, where $T = 100$ is the length of the time window, and $S = 4$ is the number of sensor channels. Among these 34,339 time windows, only $2,281$ are positive samples. Based on these numbers, as expected, one must deal with a notably imbalanced data set in which the negative class represents 93.35% of the whole data set.

3.3.3 Feature extraction

Feature extraction involves reducing the dimension of raw data to create an informative abstract representation for the classifier. This simplifies the training process by reducing the input data size and removing irrelevant information for the classification problem.
In the following section, a brief explanation of the feature extraction techniques used in this study is provided. Common and state-of-the-art feature extraction approaches in time-series analysis have been applied to four in-car signals: gas pedal position, speed, steering wheel position, and acceleration.
Two general types of feature extraction strategies will be discussed, and both will be applied in this study for automated accident detection:

- **Feature engineering:** The most commonly used feature extraction approach involves traditional feature engineering, referred to as HC in the following sections. HC features have been employed for decades and continue to be a powerful tool when combined with ML classifiers. Traditionally, HC features are crafted based on expert knowledge of the data, which may not always be available. In such cases, simple statistical attributes computed on the time-series data are commonly used, and they have demonstrated good performance in practice despite their simplicity. Feature engineering based on heuristic rules found in previous literature, as seen in studies like [74, 75], was also considered and tested in our experiments. However, due to insufficient performance, the method and results are not presented in this work.

- **Feature learning:** The second category of feature extraction approaches is feature learning, which involves the automated learning of features using deep neural networks (DNNs). An artificial neural network (ANN) is composed of a series of parametric functions represented by layers. ANNs typically consist of three layers: input, hidden, and output. A deep neural network DNN is an ANN with at least two hidden layers. Each layer comprises multiple neurons, which are simple non-linear computational units that produce a single value based on several inputs. In the general case, layer l_i with $i \in \{1 \dots \mathrm{L}\}$ takes the output of the previous layer l_{i-1} as input and applies a non-linear function to compute its own output. The last layer typically estimates class probabilities associated with the input data and uses a softmax activation layer with a number of neurons equal to the number of classes.

  DNNs have demonstrated exceptional feature learning capabilities, particularly in image classification tasks. Since our evaluation framework maintains a fixed classifier, a similar approach to that used by Li et al. [60] and Girshick et al. [103] was adopted to utilize DNNs as feature extractors. The DNNs were initially trained in a supervised manner using a softmax layer. Subsequently, the softmax layer was removed to allow the DNN to output feature vectors, which were then used to train the fixed classifier in the presented framework.

A brief explanation of the abovementioned feature extraction methods chosen for this study follows.

Feature extraction based on handcrafted features (HC)

Traditional HC feature extraction, which has been in use for decades, still proves to be a robust baseline when combined with ML classifiers, owing to its simple setup. HC features are derived from straightforward statistical attributes such as minimum, maximum, percentiles, as well as more complex descriptors like those associated with the frequency domain, computed using the Fourier transform of the signals. In the context of HC features, 18 statistical values are commonly employed in various application domains dealing with time-series data [60, 104]. These features encompass calculations based on either the time-series data or their corresponding power spectrum, as indicated in Table 3.2. Each of these features is computed individually for every sensor channel. Subsequently, the features extracted from all channels are concatenated to construct a feature vector with a total of $18 \times 4 = 72$ elements.

Table 3.2: List of the handcrafted features used in this study. Each feature is computed on each sensor channel independently.

Handcrafted Features		
Maximum	Average	Auto-correlation
Minimum	Skewness	First-order mean
Percentile 20	Kurtosis	Second-order mean
Percentile 50	Interquartile	Standard-deviation
Percentile 80	Zero-crossing	Norm of the first-order mean
Spectral entropy	Spectral energy	Norm of the second-order mean

To enhance classification efficiency, mitigate the risk of overfitting (reducing the chances of making decisions based on noise), and gain more meaningful insights into the relevant features, this study employs a feature selection approach following HC feature extraction. As outlined by Tang et al. [105], three primary categories of feature selection approaches exist:

- **Filter-based feature selection:** Filter-based feature selection approaches evaluate features without employing any classification algorithms. A typical filter algorithm comprises two steps. In the first step, it ranks features based on specific criteria. In the second step, the features with the highest rankings are selected to create classification models.

- **Wrapper-based feature selection:** Wrapper-based models utilize a specific classifier to assess the quality of the selected features. A typical wrapper model operates in three steps: *I.* Searching for a subset of features. *II.* Evaluating the selected subset of features based on the classifier's performance. *III.* Repeating steps *I* and *II* until the desired performance is achieved.

- **Embedded-based feature selection:** that integrate feature selection with classifier construction. They offer the advantages of wrapper-based models, as they involve interactions with the classification model, while also being

computationally less intensive compared to wrapper methods. Embedded feature selection approaches consider classifier biases and incorporate them, often through regularization models [106].

In this study, three popular feature selection protocols from the filter-based and wrapper-based methods were tested: ReliefF and Fisher score from the filter-based category, and Recursive Feature Elimination (RFE) from the wrapper-based category [107]. Among these three methods, only RFE demonstrated significant improvements in performance. Therefore, only the results of RFE are reported in this work.
RFE feature selection is a wrapper-based approach that evaluates various feature sets by training and assessing a classifier for each set and comparing their classification performance. Each wrapper approach proposes a strategy to select sets of input features to test, avoiding the need to test all configurations, which would be computationally expensive. RFE begins with using all features as a complete subset and training the classifier. It then eliminates features iteratively by considering smaller and smaller feature sets. In this study, features were eliminated based on feature importance scores returned by the classifier I selected (random forest and support vector machine). The step size (denoted as n) and the number of features to eliminate are determined by the lowest score. This procedure is repeated recursively until the desired number of remaining features or the desired level of performance is achieved.

Feature learning based on multi-layer-perceptron (MLP)

An MLP (Multi-Layer Perceptron) is the simplest and most traditional architecture for deep learning models. This architecture is also known as a fully connected network because neurons in layer l_i are connected to every neuron in layer l_{i-1} for $i \in [2, L]$, where L is the number of layers. Each connection has associated parameters called weights. Therefore, the connection between two consecutive layers of an MLP can be represented by the following equation:

$$\boldsymbol{x}_{l_i} = f(\boldsymbol{W}_{l_i} \times \boldsymbol{x}_{l_{i-1}} + \boldsymbol{b}_{l_i}) \tag{3.1}$$

With $n^{(i)} \in \mathbb{N}^*$ number of neurons in layer l_i, $\boldsymbol{x}_{l_i} \in \mathbb{R}^{n^{(i)}}$, $\boldsymbol{W}_{l_i} \in \mathbb{R}^{n^{(i)} \times n^{(i-1)}}$, $\boldsymbol{b}_{l_i} \in \mathbb{R}^{n^{(i)}}$, $\boldsymbol{W}_{l_i}$ being the matrix of weights connecting the neurons of layer l_{i-1} to layer l_i, $\boldsymbol{b}_{l_i}$ the vector of biases in layer l_i, $\boldsymbol{x}_{l_i}$ the output of layer l_i and f the activation function. Figure 3.3 illustrates the schematic of the MLP network used in this study.

Feature learning based on convolutional neural networks (CNN)

Convolutional Neural Networks (CNN) primarily consist of convolution layers and pooling layers. Some deep learning architectures also include batch normalization layers (see Figure 3.4). CNNs have been successfully applied in image recognition [108, 109], various natural language processing tasks [110, 111], and time-series analysis [112]. The general form of using a convolution for the centered time stamp t is given in the following equation:

$$\forall t \in [1, T], c_t = f(\boldsymbol{\omega} * \boldsymbol{x}_{t-l/2:t+l/2} + \boldsymbol{b}) \tag{3.2}$$

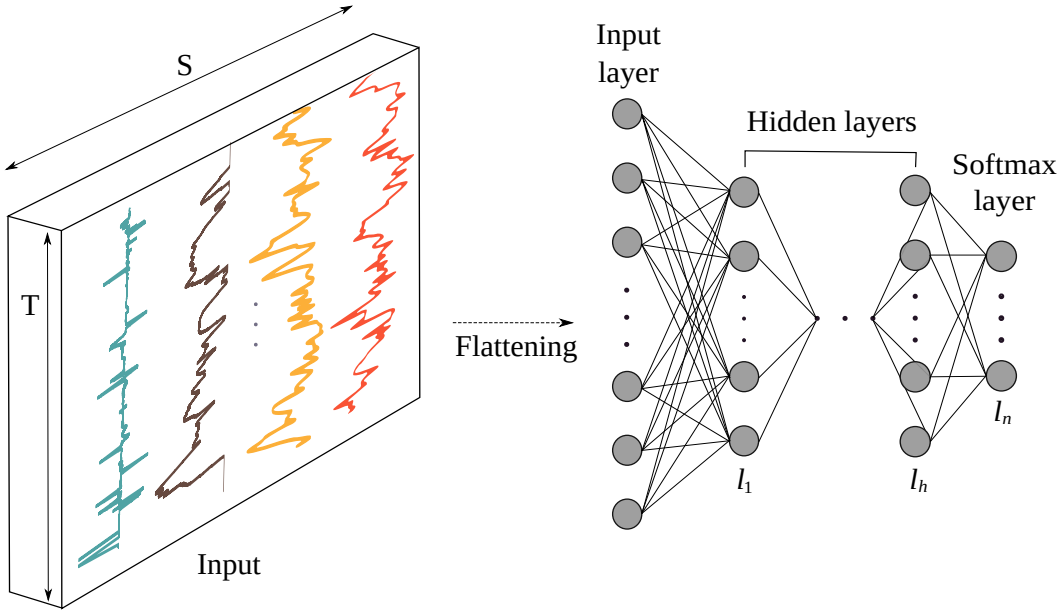


Figure 3.3: Architecture of a MLP model for accident detection with h hidden layers, n number of classes and S number of sensor channels. Input data are first flattened into a $(T \times S)$-dimensional vector and and then fed to the hidden layers. All layers are fully connected.

where $*$ designates the convolution product, c_t the result of the convolution at time t, f the activation function, $\boldsymbol{x}$ a 1D input, $\boldsymbol{\omega}$ the convolutional filter of length l and $\boldsymbol{b}$ a bias parameter. A convolution can be seen as applying and sliding a filter over a time-series or in other words as a generic non-linear transformation of an input vector $\boldsymbol{x}$. For instance, if convoluting a time-series with a filter of length 3 with values equal to $[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$ is the equivalent of applying a moving average with a sliding window of size 3.

The convolutional maps obtained after applying several convolutional kernels are typically used as input for a pooling layer, which can be either local or global. Local pooling operations, such as averaging or taking the maximum value in a sliding window, are applied to downsample the input of the layer. In contrast, global pooling involves downsampling the entire input time dimension, resulting in a single output value. To aid the network's convergence during training, a normalization layer is sometimes added. Two common normalization layers used in CNNs for time-series analysis are batch normalization [113] and instance normalization [114]. When used for classification, CNNs are often connected to a classification MLP, where the last layer is a softmax layer, providing a distribution score over the class variables in the data set.

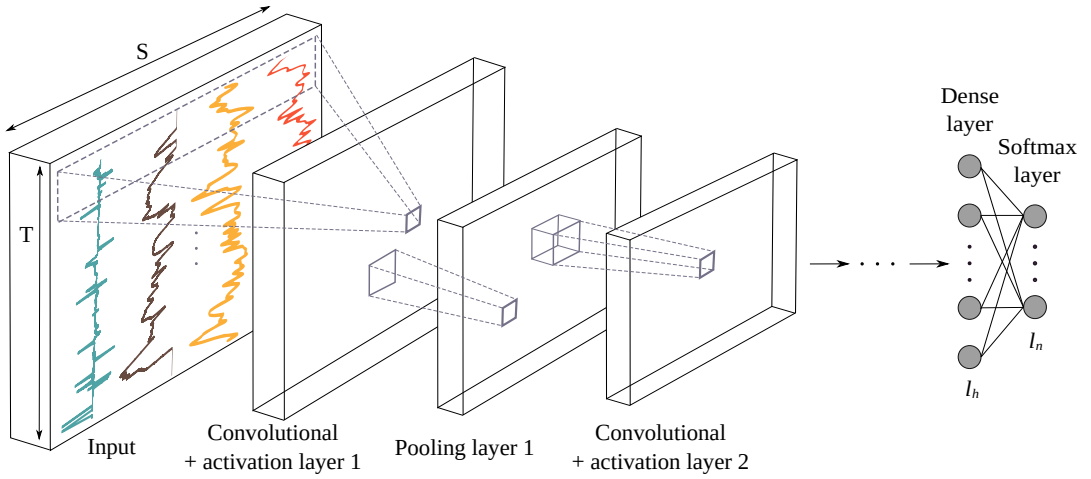


Figure 3.4: Architecture of a CNN model for accident detection. The designations n, h and S are the number of classes, layers and sensor channels, respectively. Convolutional layers apply convolution products on all convolution maps of the previous layer. Pooling layers then downsample the convolutional output and pass it to the next convolutional layer.

Feature learning based on long short-term memory (LSTM)

Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Network (RNN), which are a special type of DNNs that use loops and connections between nodes to handle temporal sequences and retain dynamic temporal information throughout the network.
LSTM cells include internal mechanisms called gates, which manage the flow of information over time by storing it in an internal memory, updating, outputting, or erasing this internal state based on their input and the state from the previous time step [115]. These gate operations can be described as follows:

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_{\boldsymbol{f}}) \tag{3.3}$$

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_{\boldsymbol{i}}) \tag{3.4}$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_{\boldsymbol{o}}) \tag{3.5}$$

$$\boldsymbol{m}_t = \boldsymbol{f}_t \otimes \boldsymbol{m}_{t-1} + \boldsymbol{i}_t \otimes \sigma(\boldsymbol{W}_c \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_C) \tag{3.6}$$

where x_t is the input vector to the LSTM cell and h_{t-1} the hidden state vector also known as the output vector of the LSTM cell. $\boldsymbol{f}_t$, $\boldsymbol{i}_t$ and $\boldsymbol{o}_t$ represent forget, input and output gates, respectively. These gates have their own weights ($\boldsymbol{W}_*$), bias ($\boldsymbol{b}_*$) and activation functions (σ). The functionality of an input, output and forget gates are used to block the input of the cell , block its output, and erase its internal memory at time t. $\boldsymbol{m}_t$ is the memory state of the cell at time t, and $\otimes$ is the element-wise multiplication of two vectors. The architecture of the RNN with LSTM layers containing LSTM cells is shown in Figure 3.5. Like all DNNs, there are different architectural variations of LSTM layers, and the last layer is followed by dense and softmax layers.

Feature learning based on an autoencoder (AE)

Autoencoders, a special type of DNNs, are trained to replicate input data at their output by utilizing a loss function like mean squared errors in an unsupervised manner. These networks perform dimensionality reduction by initially projecting input data into an embedding space with a smaller dimensionality than the input space using an encoder. Subsequently, they decompress the embedding to closely match the original input using a decoder. It's important to note that autoencoders always have the same number of inputs as outputs. The architecture of an autoencoder is schematically represented in Figure 3.6. For feature extraction, autoencoders are initially trained in an unsupervised manner to reconstruct their inputs on the output layer. Afterward, the decoder is removed, and the encoder is utilized to output feature vectors.

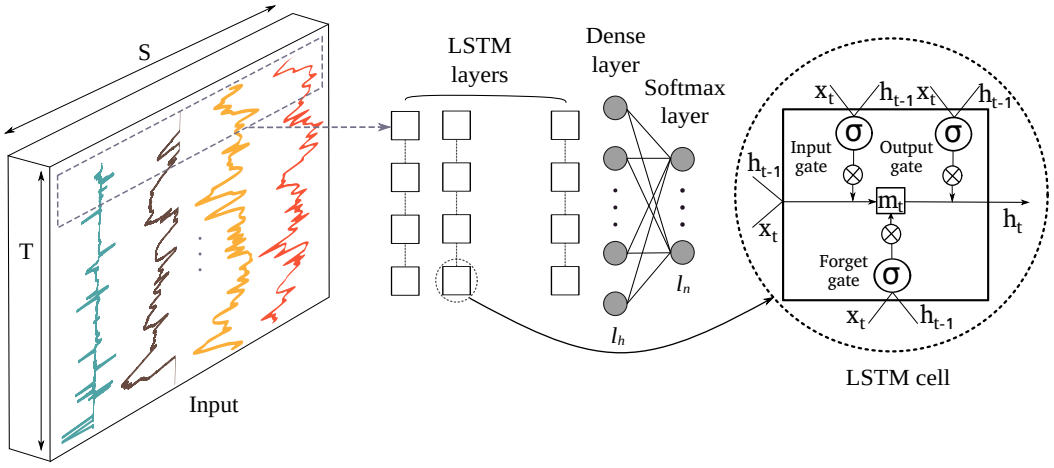


Figure 3.5: Architecture of a LSTM network for accident detection . The designations n, h and S are the number of classes, layers and sensor channels, respectively. The cells in the LSTM layers have one input, forget and output gates. $\boldsymbol{x}_t$, $\boldsymbol{m}_t$ and $\boldsymbol{h}_t$ refer to the cell input, memory, and output at time t, respectively, and $\boldsymbol{\sigma}$ designates the activation function.

3.3.4 Classification

The final step in the ML framework is classification, which involves training a model to predict class labels (categories) based on an input feature vector associated with a specific data segment. The classifier establishes boundaries between different classes within the feature space. Various popular classifiers found in past literature include Support Vector Machine (SVM) [13], Random Forest (RF) [17], k-Nearest Neighbors (kNN) [116], Decision Tree [117], and others.
In this study, two classifiers have been chosen due to their high performance and their ability to mitigate overfitting in the case of high-dimensional data. These classifiers are a soft-margin SVM (C-SVM) with a radial basis function (RBF) kernel and a Random Forest (RF) classifier [60, 118, 119].

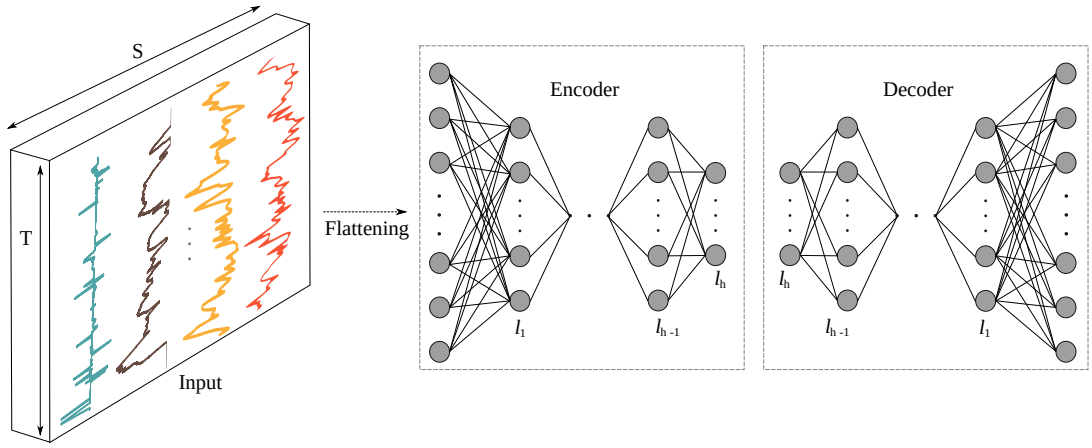


Figure 3.6: Autoencoder's architecture used for accident detection consists of encoder and decoder sections. The encoded layer is used as features for the feature learning purposes; S is the number of sensor channels and h denotes the number of hidden layers in both the encoder and decoder.

The Support Vector Machine (SVM) is a highly effective linear classifier that can also handle non-linear cases through the use of the kernel trick. The C-SVM, or soft-margin SVM, is a variation of the SVM that permits misclassifications during training to prevent overfitting. It achieves this by regulating the soft-margin parameter, denoted as C, which controls the extent to which misclassifications are allowed. When training an SVM with the radial basis function (RBF) kernel, two essential parameters need to be considered: C and *gamma*. These parameters play a critical role in the SVM's performance and effectiveness.
The parameter C plays a crucial role in determining the trade-off between misclassifying training examples and achieving a simpler decision surface in the Support Vector Machine (SVM). Similarly, the parameter *gamma* is of great significance for the SVM's performance. The choice of appropriate C and *gamma* values is essential for obtaining good results with the SVM.
In this study, a grid search was conducted to identify the optimal values of C and *gamma* for each feature set derived from the methods mentioned above. The Random Forest (RF) is another widely used classifier that employs ensemble techniques with decision trees to mitigate overfitting. The primary hyperparameter in RF is the number of trees, denoted as T. Like the SVM parameters, the number of RF trees, T, was optimized through a one-dimensional grid search for each feature set individually. This optimization process ensures that the classifiers are well-tuned for the specific features they are trained on.
It's important to highlight that DNNs are typically trained with a classification layer as part of the network architecture. To utilize the previously mentioned classifiers on features extracted from DNNs, the following process was required:

1. Initially, the DNN model was trained in the usual manner, including the classification layer (softmax) for class prediction.

2. After training, the classification layer (softmax) was removed from the model,

and the output of the penultimate layer was used as an input feature for the classifier. This penultimate layer's output serves as the feature representation for classification.

For feature extraction based on Autoencoders (AE), the feature representation is directly obtained from the output of the encoder network after training the complete AE model.

3.4 Experiments and results

This section outlines the implementation details of the proposed ML-based framework for automated accident detection using multimodal in-car sensors. It also provides information on the evaluation setup and presents the results.
All the implementations in this chapter of the dissertation were coded in Python. Feature learning approaches using DNNs were implemented using the *Keras 2.1.0* framework with a *Tensorflow 1.14.0* backend, *scikit-learn 0.21.3*, and trained using the ADADELTA optimizer with default parameters (initial learning rate of one) for 50 epochs, with a batch size ranging from 100 to 1000.
To comprehensively evaluate the model's performance across the entire data set, a K-fold cross-validation was applied, with K set to 5. In each training run, one of the five partitions was selected as the test set, and the remaining partitions were used for training. The details of the experiments for each feature extraction algorithm are described as follows:

HC: The HC features consisted of 15 statistical values directly computed on the time-series and 3 frequency-related on their power spectrum were computed on each sensor individually and concatenated together. Then RFE feature selection with an elimination size of three was applied.

MLP: The MLP architecture used in this study contained three dense layers and REctified Linear Units (RELU) activation. MLP usually takes 1D inputs only, therefore a flattened layer was used to convert the 2D input to 1D. According to the recommendations of [113, 60], a batch normalization layer was placed directly after the network input to improve results. Three fully connected dense layers with RELU activation function, containing 2000 neurons each, and a final softmax layer built up the MLP network used for this study (see Table 3.3). In Table 3.3, the values for the hyperparameters used for feature learning approaches in this study are shown. Optimizing the hyperparameters of DNNs is an important and difficult topic. Optimal parameters for mentioned models were chosen after testing several manually selected configurations. Manual hyperparameter selection is the default approach in the literature due to the absence of other more elaborated high performing approaches.

CNN: As listed in Table 3.3, the CNN layout consisted of three blocks of batch normalization and a convolutional layer with RELU activation followed by dense and pooling layers. The CNN design was based on [60] with some modifications, including a reduction in the size of the convolutional kernels

and an increase pooling window-size, while keeping the amount of kernels the same for each block.

LSTM: The values of all hyperparameters for the LSTM architecture are provided in Table 3.3. Like other ANNs used in this work, a batch normalization layer was added at the beginning and a dense and softmax layers at the end of the network. The gate activation used in the LSTM cells is a sigmond function, and in the dense layers, a tangent activation function was used.

AE: The AE architecture consisted of simple dense layers (three dense layers for the Encoder and then three for the Decoder designed as a mirror), with ReLU for the activation function. Different numbers of dense layers were tested, and the one achieving the best performance is presented in Table 3.3.

Table 3.3: Hyperparameters of the ANN models on the SHRP2 data set.

Model	**Parameter**	**Value/ Type**
MLP	. # Dense layers	3
	. # Neurons in each layer	2000
	. Activation function	ReLU
CNN	. # Conv. blocks	3
	. Conv. kernel size for blocks 1, 2 and 3	(5, 1), (4, 1), (3, 1)
	. # Conv.kernels in each block	50
	. Pool size for blocks 1, 2 and 3	(2, 1), (3, 1), (4, 1)
	. # Neurons in the dense layer	1000
	. Activation function for the Conv. blocks	Tanh
	. Activation function for the dense layer	ReLU
LSTM	. # LSTM layers	2
	. # Output dimensions for each LSTM cell	600
	. # Neurons in the dense layer	512
	. Activation function for the dense layer	ReLU
AE	. # Encoder dense layers	3
	. # Neurons in layers 1, 2 and 3	5000, 3000, 1000
	. Activation function	ReLU

The hyperparameters for the above mentioned methods are depicted manually. The evaluation of proposed feature extraction framework is based on three different metrics, average $F1$ score, overall accuracy and weighted $F1$ score. $F1$ score is the harmonic mean of precision and recall, and average $F1$ score is the mean value of each class $F1$ scores. As mentioned in section 3.3.1, the used data set was strongly imbalanced with only 6.65% positive samples, and both accuracy and weighted $F1$ score are very biased in case of an unbalanced data set. For this reason, the average $F1$ score, which is the mean value of each class $F1$ score, is considered as the main evaluation metric due to its ability to take class imbalance into account [120].

The results of the aforementioned feature extraction approaches with both SVM and RF classification methods are provided in Tables 3.4 and 3.5, respectively.

Table 3.4: SVM classification evaluation metrics (in percent) of different tested feature extraction models on the SHRP2 data set.

Methods	**Accuracy**	**Weighted F1 Score**	**Average F1 Score**
HC	94.34	92.99	66.56
MLP	83.60	82.30	75.00
CNN	**85.72**	**84.90**	**79.10**
LSTM	76.81	72.01	57.90
AE	83.40	82.40	75.50

Three main observations can be drawn from Tables 3.4 and 3.5. First, the choice of the classifier impacts the final classification performance. RF improved the average $F1$ score of HC feature extraction to 71.78%, and a notable improvement of the RF with almost 10% more for average $F1$ score is for LSTM feature learning. Second, deep feature learning outperforms feature engineering. In particular, CNN surpasses other methods by a remarkable average $F1$ score of 79.10% and 78.39% for SVM and RF, respectively. Finally, the quite decent performances with unsupervised deep feature learning (AE) when compared to supervised feature learning, is a remarkable observation of this study.

Table 3.5: RF classification performance metrics (in percent) of different feature extraction models of five cross on the SHRP2 data set.

Methods	**Accuracy**	**Weighted *F*1 Score**	**Average *F*1 Score**
HC	94.97	93.95	71.78
MLP	84.06	83.57	77.47
CNN	85.72	84.19	78.39
LSTM	78.00	76.61	67.22
AE	84.22	83.74	77.67

3.5 Discussion

The experiment on applying feature extraction approaches on a SHRP2 crash data set reveals the following points: First, all feature learning approaches except LSTM with both classifiers outperform the feature engineering. LSTMs are usually prone to high computation time and are difficult to tune. Another reason for the poor performances of LSTM is the length of the time horizon ($T = 100$) of the input samples being too long. This happens quite often when dealing with time-series and is one of the reasons CNNs are preferred over RNNs for time-series processing in the literature.

CNN feature learning obtains a stable top performance with both classifiers. Obtained results are consistent with the literature, which seems to indicate that CNNs

are the most reliable architecture for time-series classification [121, 122]. To understand the difference between HC features and features created by deep feature learning, the best features extracted by both approaches were analyzed in this work. For HC, it is possible to use the ranking of the RFE algorithm which was applied for feature selection.
Figure 3.7 shows the RFE on HC features for the five cross-validation folds. In these figures, the x-axis is the RFE eliminating steps, and the y-axis shows eighteen HC features of each sensor channel. The RFE method removes features iteratively, starting with the ones with the least impact on the final classification performance. As can be seen in all five Figures, the HC features extracted by gas and acceleration sensors are the ones that are eliminated last, meaning that features from these two sensor channels are the most important HC features selected by RFE.

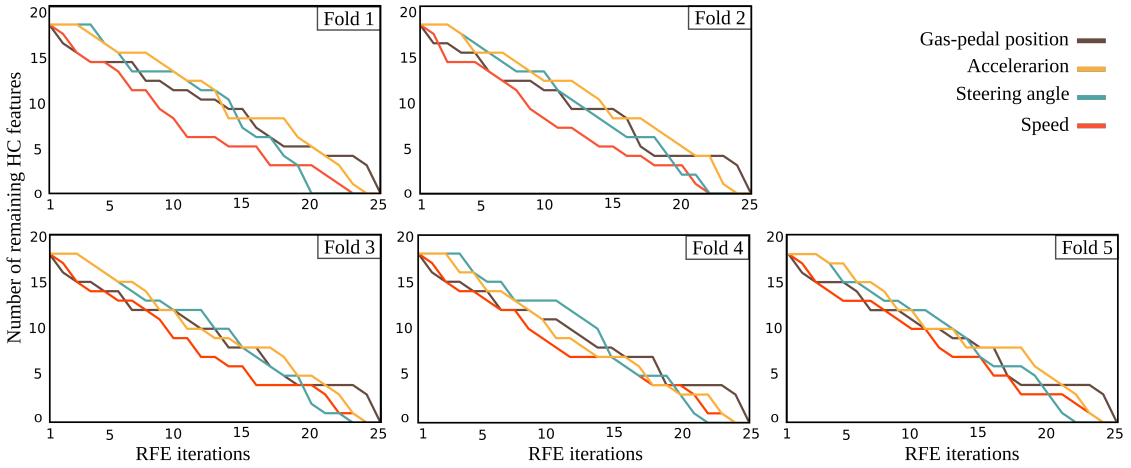


Figure 3.7: RFE results per sensor channel for five cross-validation folds. The y-axis shows the number of remaining features from each sensor, and the x-axis shows the step by step removal of features from each sensor.

For DNNs, finding the best features is more challenging since features learned by DNNs are hard to interpret. Instead, I decided to use an approach based on the Jacobian matrix of the model to determine which sensor channels are the most important and whether this matches the observation of the HC results. In case of considering a trained DNN as an approximation of a multi-input/multi-output function $f : \mathcal{R}^{L \times S} \rightarrow R^{C}$, where L is the length of a multichannel segment $\boldsymbol{X}$ belonging to the target data set $\mathcal{X}_T$, S is the number of channels, and C_T is the number of classes, it is possible to compute the Jacobian matrix of function f. Each Jacobian value $J_{c,l,s}(\boldsymbol{X})$ in the Jacobian Matrix, represents the importance of x_{ls}—the value at the lth time point ($1 \leq l \leq L$) of the sth sensor—on the predictive function for the cth class. It can therefore be used to indicate which parts of the input have the most impact on the output classification score. With this, it is possible to propose a Jacobian score similar to [123] that would indicate how important to the final classification score a specific sensor channel is.
A channel-wise Jacobian score for $\omega_s(\boldsymbol{X})$ as the average of absolute $J_{c,l,s}(\boldsymbol{X})$ over all the L time points and all the C_T classes is,

$$\omega_s(\boldsymbol{X}) = \frac{1}{C_T}\frac{1}{L}\sum_{c=1}^{C_T}\sum_{l=1}^{L} \mid J_{c,l,s}(\boldsymbol{X}) \mid \tag{3.7}$$

In addition, as described in [123], the global channel-wise Jacobian score Ω_s is averaging $\omega_s(\boldsymbol{X})$ over all examples $\boldsymbol{X}_T$ in the data set, where $|(\mathcal{X}_T)|$ refers to the cardinality of $\mathcal{X}_T$:

$$\Omega_s = \frac{1}{|(\mathcal{X}_T)|}\sum_{\boldsymbol{X}\in\mathcal{X}_T} \omega_s(\boldsymbol{X}) \tag{3.8}$$

A high Ω_s indicates a high importance of the sensor channel for the classification problem. Figure 3.8 shows the Jacobian scores for the four input sensor channels (speed, gas position, steering-angle, acceleration) and for each of the five CNNs trained on each cross-validation fold. According to the Jacobian scores of these sensors, steering-angle has the highest impact on the result. Speed and acceleration are the second most important input signals. Finally, this study shows that gas-position was consistently found to be considered as the least useful channel by CNN, which is opposite to feature extraction with HC that obtained its best features from this channel. This would indicate that the features learned by HC and CNN are different in nature and might be complementary. Additionally, I present a hypothesis regarding why steering-angle information was regarded as useful for the classification problem by CNNs and not by HC. The steering-angle signal, compared to the other sensor channels, is characterized by strong and rapid fluctuations that might contain valuable frequency-based information for the considered classification problem. Contrary to CNN, this information might not have been captured well enough by the HC features that were extracted.

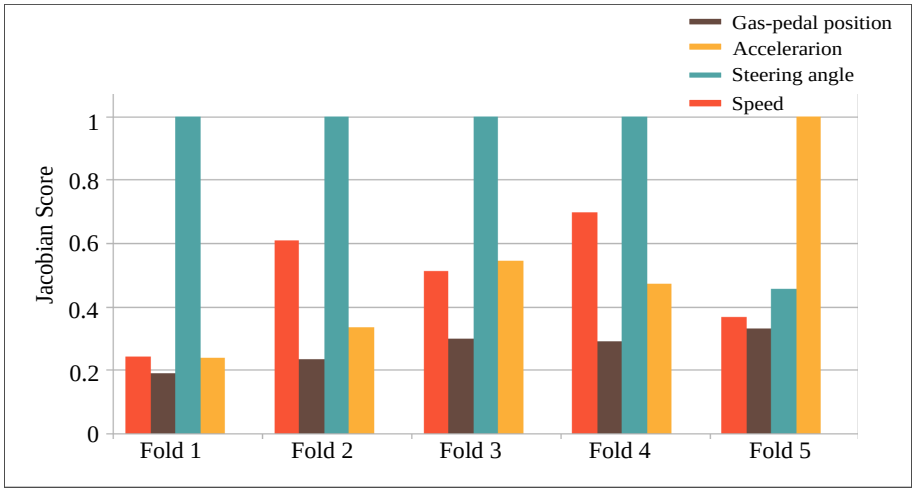


Figure 3.8: Jacobian score for CNN of five cross-validation folds of four signals. A higher Jacobian score indicates that the sensor channel contributed to the learning of more discriminative features. The y axis represents the normalized magnitude of the Jacobian score for each sensor channel.

Analysis of the feature importance results led us to combine CNN features with the optimal ranked HC features to improve the results. The same steps were used to calculate the features as described in section 3.3.3. Then the RFE was used to rank all features and keep only the three best ones. These features were then appended to the 1000 features of CNN in section 3.3.3. Combined CNN features improved the average $F1$ score from 79.10% to 80.12% for a SVM and from 78.39% to 79.10% for RF classifier which seems to confirm expected hypothesis that both features contain complementary information.

3.6 Summary

This chapter introduced a framework for accident detection using basic in-car sensors from the SHRP2 naturalistic data set, employing various machine learning approaches. The study tested state-of-the-art feature extraction methods, including traditional manual feature extraction and feature learning, in combination with two classifiers.
Among the approaches tested, CNN features combined with an SVM classifier outperformed all others. This combination achieved an accuracy of 85.72%, a weighted $F1$ score of 84.9%, and an average F1-score of 79.10%. These results are highly promising, especially considering that the data set is based on naturalistic accidents, with very few samples of severe accidents that can be recognized by only four basic in-car sensors.
Additionally, the interpretability studies demonstrated that traditional manual feature engineering and DNNs were extracting their optimal features from different sensor channels. This indicates that a combination of these approaches could be highly effective due to their complementary nature.
The study has several limitations. Firstly, it relies on a single data set due to the scarcity of data for this specific application, which limits the generalizability of the results. Secondly, comparing these findings to other studies is challenging due to variations in accident definitions and the types of input sensor channels used.
In conclusion, some key takeaways from this study include:

- Promising results can be achieved with machine learning for accident detection using fundamental in-car sensor data.
- A deep learning feature extraction method performs better in comparison with HC, and unsupervised feature extraction remarkably achieves the second best performance score.

Future work will encompass four main areas. First, expanding the range of tested sensor channels and exploring alternative learning algorithms to enhance classification performance. Incorporating additional sensor modalities like lateral acceleration and yaw rate, which are common in all cars, and analyzing their impact on detection performance is a potential avenue for further investigation. Second, improving accident detection results by employing more advanced feature selection algorithms, particularly embedded feature selection methods [124] designed for unbalanced data

sets. Third, considering the promising results of unsupervised deep feature learning (Autoencoders), further research into unsupervised feature learning techniques is recommended. Lastly, to address the challenge of limited labeled data, investigating transfer learning techniques [125] as a potential solution is worth exploring. These directions can contribute to the ongoing development and enhancement of accident detection systems based on in-car sensor data.

Chapter 4

Conclusion

4.1 Summary

Driving behavior analysis plays a pivotal role in advancing road safety, optimizing transportation systems, enhancing driver education, and enabling the development of advanced technologies for safer and more efficient road environments. Machine learning, in particular, plays a crucial part in driving behavior analysis by facilitating the extraction of valuable insights from large data sets.
Leveraging the capabilities of machine learning, driving behavior analysis can harness advanced data processing, predictive modeling, and pattern recognition, ultimately leading to improved road safety, personalized feedback for drivers, and more informed decision-making for various stakeholders involved in transportation and road management.
However, challenges persist in the ML-based driving behavior literature, primarily due to the lack of affordable and reliable data modalities and an insufficient amount of labeled data. Despite these challenges, this work has made contributions to the domain, particularly in the areas of anomaly detection and accident detection, using state-of-the-art ML approaches.
To summarize, this work addresses the questions raised in section 1.1.1 and provides valuable insights and solutions in the field of driving behavior analysis.

- **Utilizing primary in-car sensors using ML approaches:** Is it possible to utilize primary in-car sensors using ML approaches to detect abnormal driving patterns? The first section of chapter 2 of this thesis has shown that this is indeed possible. In-car sensors, including those measuring speed, brake, throttle, and steering-angle, offer valuable data that can be used to identify deviations from typical driving behavior.

  Machine learning techniques are well-suited for processing and analyzing this sensor data, enabling the detection of abnormal driving patterns. There is a wide range of ML algorithms that can be applied, from traditional methods such as support vector machines (SVMs) and cluster-based models to more advanced techniques like recurrent neural networks (RNNs). The choice of algorithm should take into account the availability of labeled data and the volume of data to be analyzed.

In summary, the integration of primary in-car sensors and ML approaches is a feasible and effective way to identify abnormal driving patterns, contributing to enhanced road safety and driver behavior analysis.

Considering the difficulty of adequately training and interpreting unsupervised deep learning models, is there a benefit to using them for anomaly detection as opposed to other traditional machine learning models? Indeed, there are benefits to using unsupervised deep learning models for anomaly detection in comparison to traditional machine learning models, as discussed in chapter 2 of this thesis. Some of these advantages include the ability of deep learning models to automatically learn intricate representations from data, handle high-dimensional and unstructured data effectively, and potentially capture subtle, non-linear patterns that might be challenging for traditional methods to discern.

However, it is important to acknowledge that training and interpreting unsupervised deep learning models can be more challenging and resource-intensive, and they may require larger data sets. The complexity of deep learning models can also make them less interpretable than some traditional ML techniques.

The choice between unsupervised deep learning models and traditional ML models for anomaly detection should be based on the specific needs and constraints of the application. Factors such as the availability of labeled data, the nature and complexity of the data, computational resources, and the interpretability of the model all play a role in making an informed decision regarding which approach to use.

Regarding the lack of labeled driving patterns, can applying the proposed solution to the benchmark driving data set be worthwhile? Utilizing the proposed solution on a benchmark driving data set can indeed be a valuable approach, particularly when labeled driving patterns are scarce or not readily accessible. As demonstrated in section 2.5 of this work, the application of the proposed PRC framework to a naturalistic data set of hazardous driving events has shown significant success. This framework provides an effective means to detect and analyze abnormal driving behavior, making it a promising tool for improving road safety and driving behavior analysis.

- **Detecting real-world accidents based on primary in-car sensor data:** Is it possible to detect real-world accidents based on primary in-car sensor data? Detecting real-world accidents using primary in-car sensor data is indeed feasible. These sensors, including speed, throttle, brake, steering-angle, offer valuable information that can be harnessed to identify patterns and events indicative of accidents or hazardous situations.

  However, it is crucial to acknowledge that accident detection relying solely on primary in-car sensor data may come with limitations. The performance of detection can be affected by factors like sensor quality and reliability, the availability and quality of labeled training data, and the specific context and conditions in which accidents occur. Therefore, while primary in-car sensors

can provide valuable insights into accident detection, it's important to consider these factors and their potential impact on the accuracy of detection.

What is the best feature extraction method for accident detection? What features contribute more to the classification result? In section 3.3, various feature extraction methods, including traditional manual feature extraction and feature learning, were applied alongside two classifiers. Among the tested approaches, CNN features in combination with an SVM classifier achieved the highest performance, surpassing all others. This approach achieved an accuracy of 85.72%, a weighted $F1$ score of 84.9%, and an average $F1$ score of 79.10%. These results are promising, given that the data set used in this study is based on real accidents, and the recognition of severe accidents relied solely on data from four basic in-car sensors. Furthermore, interpretability studies demonstrated that the combined HC and DNN approaches extracted optimal features from different sensor channels, indicating their potential for effective combination due to their complementarity. Combined HC and DNN features improved the average $F1$ score from 79.10% to 80.12% for the SVM classifier.

Based on the Jacobian scores of the sensors used, the steering-angle sensor has the most significant impact on the results. Speed and acceleration sensors come next in terms of importance. Interestingly, the results presented in section 3.4 highlight that the gas pedal position sensor was consistently considered the least informative channel by CNN, which contrasts with the feature extraction using HC, where it yielded the best features. This suggests that the features learned by HC and CNN have different characteristics and may be complementary. As a result, a combination of CNN features with the top-ranked HC features was explored to enhance the results.

It is important to emphasize that the research presented in this thesis represents the pioneering study in the field of ML-based accident detection using primary in-car network data. This study stands as a unique and innovative research endeavor, focusing on the detection of real driving accidents using data from the most accessible and cost-effective sources within vehicles.

4.2 Limitations

This thesis delivered a comprehensive investigation into the utilization of cutting-edge ML techniques applied to primary in-car sensor data for the detection of anomalous driving behaviors. Furthermore, it introduced an extensive ML framework, based on the PRC, for the purpose of accident detection using primary in-car data. The framework was then utilized to compare state-of-the-art ML feature extraction techniques that are relevant to in-car sensor data for accident detection, specifically based on the SHRP2 NDS crash data set.
Despite yielding promising results, the studies presented in this work have several limitations that warrant further exploration. The most significant limitations are summarized as follows:

- **The finite scope of the ML-based driving behavior study using primary in-car sensor data:** The lack of ground truth in ML-based driving behavior studies is attributed to various factors. Machine learning can aid in the analysis of driving behaviors by learning from extensive data sets and uncovering patterns and correlations. However, establishing a definitive ground truth for driving behavior is challenging, as outlined in section 1.1.1.

- **Imbalanced data set:** Driving behavior studies often face challenges related to data scarcity, particularly in the case of naturalistic data. However, the lack of positive examples is another significant issue in this field when data is available. For many machine learning tasks, having both positive and negative behavior examples is essential. In the context of driving, positive examples refer to dangerous or unacceptable behaviors, which can be challenging to collect due to ethical and safety concerns.

- **Data acquisition challenges:** Collecting comprehensive and accurate driving data is often an expensive and time-consuming process. It encompasses various aspects, including vehicle dynamics, driver reactions, and environmental conditions. Moreover, it necessitates specialized equipment, such as in-vehicle sensors, cameras, or telematics devices, and the participation of willing individuals, which contributes to the complexity and cost. In the case of supervised machine learning models, each data point must be precisely labeled. However, labeling driving data can be challenging due to the subjective nature of what constitutes specific driving behaviors, and it can also be a laborious and time-consuming task.

- **Defining anomalies and interpreting the results:** Defining anomalies and interpreting results in driving behavior analysis can be challenging. What may be considered an anomaly in one context might be normal behavior in another. Advanced machine learning models, such as neural networks, are often referred to as black boxes because their internal operations can be challenging to interpret. While these models can accurately detect anomalies, explaining why a specific behavior was identified as an anomaly can be difficult. This lack of interpretability can hinder researchers and practitioners in trusting and comprehending the results.

- **The complexity of events such as accidents:** Establishing a clear definition of a car accident is complex due to various factors. Accidents can vary widely in terms of speed, impact direction, types of vehicles involved, and other variables, all of which can significantly influence the severity of an accident. What one person might consider a minor accident, another might deem severe, depending on their perspective, experience, or personal bias. This subjectivity makes it challenging to label and classify accidents accurately.

4.3 Future work

The following directions could be explored to expand the scope of the thesis contents and address some of the limitations previously mentioned:

- **Expanding the number of sensor channels:** Expanding the number of sensor channels tested in this work is essential. Exploring different combinations of sensor channels and utilizing various learning algorithms may lead to enhanced classification performance. Additionally, including additional sensor modalities such as lateral acceleration and yaw rate, which are readily available in most cars, and examining their impact on the detection performance is a promising avenue for future research.

- **Further investigation in unsupervised approaches:** Unsupervised approaches have shown remarkable results in both anomaly detection and accident detection in this thesis. Given these promising performances, delving deeper into the exploration of unsupervised learning techniques such as Generative Adversarial Networks (GANs) [126] in future work is highly recommended. This can help uncover more insights and potentially lead to even better results in various applications related to driving behavior analysis.

- **Using advanced feature selection approaches:** Chapter 3 of this thesis highlights the importance of feature selection methods in enhancing classification model accuracy, mitigating overfitting, and improving model robustness. It discusses both filter-based and wrapper-based feature selection methods.

  In future work, it would be valuable to explore more advanced feature selection algorithms, such as embedded feature selection methods designed for unbalanced data sets [124]. This can contribute to further improving the effectiveness of feature selection techniques, especially when dealing with imbalanced data, which is a common scenario in driving behavior analysis.

- **Bypassing insufficient labeled data:** To address the issue of limited labeled data, several techniques can be explored in future work:

  Semi-Supervised Learning: This approach combines both labeled and unlabeled data for training. It can effectively leverage a small amount of labeled data along with a large pool of unlabeled data to improve model performance [127] .

  Active Learning: Active Learning is a process where the model is initially trained on a small batch of labeled data. It then uses this initial training to make predictions on unlabeled data. The model selects the instances it is most uncertain about, and those instances are manually labeled and added to the training data. This iterative process continues until the model's performance reaches a satisfactory level [128].

  Transfer Learning: This approach can help overcome the challenge of limited labeled data by utilizing and fine-tuning a pre-trained model. The pre-trained model is typically trained on a large-scale labeled data set. When the

new task is related to the initial training task, Transfer Learning can be a powerful way to leverage the knowledge encoded in the pre-trained model [129]. These techniques have the potential to expand the scope of driving behavior analysis by making more efficient use of limited labeled data and addressing the challenges associated with data scarcity.

Bibliography

[1] Hawzhin Hozhabr Pour, Lukas Wegmeth, Alexander Kordes, Marcin Grzegorzek, and Roland Wismüller. Feature extraction and classification of sensor signals in cars based on a modified codebook approach. In *International Conference on Computer Recognition Systems*, pages 184–194. Springer, 2019.

[2] INVERS GmbH. www.invers.com/de/. Accessed on 02 24, 2022.

[3] Kimiaki Shirahama, Lukas Köping, and Marcin Grzegorzek. Codebook approach for sensor-based human activity recognition. ACM, Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct, 2016.

[4] Kardi Teknomo. K-means clustering tutorial. *Medicine*, 100(4):3, 2006.

[5] George R Terrell and David W Scott. Variable kernel density estimation. *The Annals of Statistics*, pages 1236–1265, 1992.

[6] Maurizio Filippone, Francesco Camastra, Francesco Masulli, and Stefano Rovetta. A survey of kernel and spectral methods for clustering. *Pattern recognition*, 41(1):176–190, 2008.

[7] Shun-ichi Amari and Si Wu. Improving support vector machine classifiers by modifying kernel functions. *Neural Networks*, 12(6):783–789, 1999.

[8] Thomas G Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of artificial intelligence research*, 2:263–286, 1995.

[9] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[10] Gene H Golub, Michael Heath, and Grace Wahba. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–223, 1979.

[11] Hawzhin Hozhabr Pour, Frédéric Li, Lukas Wegmeth, Christian Trense, Rafał Doniec, Marcin Grzegorzek, and Roland Wismüller. A machine learning framework for automated accident detection based on multimodal sensors in cars. *Sensors*, 22(10):3634, 2022.

[12] Xiangjie Kong, Haoran Gao, Osama Alfarraj, Qichao Ni, Chaofan Zheng, and Guojiang Shen. Huad: Hierarchical urban anomaly detection based on spatio-temporal data. *IEEE Access*, 8:26573–26582, 2020.

[13] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[14] Stephan Dreiseitl and Lucila Ohno-Machado. Logistic regression and artificial neural network classification models: a methodology review. *Journal of biomedical informatics*, 35(5-6):352–359, 2002.

[15] Irina Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

[16] Jorma Laaksonen and Erkki Oja. Classification with learning k-nearest neighbors. In *Proceedings of international conference on neural networks (ICNN'96)*, volume 3, pages 1480–1483. IEEE, 1996.

[17] Mahesh Pal. Random forest classifier for remote sensing classification. *International journal of remote sensing*, 26(1):217–222, 2005.

[18] Fionn Murtagh. Multilayer perceptrons for classification and regression. *Neurocomputing*, 2(5-6):183–197, 1991.

[19] Mark A Friedl and Carla E Brodley. Decision tree classification of land cover from remotely sensed data. *Remote sensing of environment*, 61(3):399–409, 1997.

[20] Zhaomin Chen, Chai Kiat Yeo, Bu Sung Lee, and Chiew Tong Lau. Autoencoder-based network anomaly detection. In *2018 Wireless telecommunications symposium (WTS)*, pages 1–5. IEEE, 2018.

[21] Simon Hawkins, Hongxing He, Graham Williams, and Rohan Baxter. Outlier detection using replicator neural networks. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 170–180. Springer, 2002.

[22] Milan Kumari and Sunila Godara. Comparative study of data mining classification methods in cardiovascular disease prediction. *Ijcst*, 2(2):304–308, 2011.

[23] Sagar S Nikam. A comparative study of classification techniques in data mining algorithms. *Oriental Journal of Computer Science and Technology*, 8(1):13–19, 2015.

[24] Rasheda Smith, Alan Bivens, Mark Embrechts, Chandrika Palagiri, and Boleslaw Szymanski. Clustering approaches for anomaly based intrusion detection. *Proceedings of intelligent engineering systems through artificial neural networks*, 9, 2002.

[25] Clay Spence, Lucas Parra, and Paul Sajda. Detection, synthesis and compression in mammographic image analysis with a hierarchical image probability model. In *Proceedings IEEE workshop on mathematical methods in biomedical image analysis (MMBIA 2001)*, pages 3–10. IEEE, 2001.

[26] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[27] Markos Markou and Sameer Singh. Novelty detection: a review—part 1: statistical approaches. *Signal processing*, 83(12):2481–2497, 2003.

[28] Minh Van Ly, Sujitha Martin, and Mohan M Trivedi. Driver classification and driving style recognition using inertial sensors. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 1040–1045. IEEE, 2013.

[29] Saif Al-Sultan, Ali H Al-Bayatti, and Hussein Zedan. Context-aware driver behavior detection system in intelligent transportation systems. *IEEE transactions on vehicular technology*, 62(9):4264–4275, 2013.

[30] Mingming Zhang, Chao Chen, Tianyu Wo, Tao Xie, Md Zakirul Alam Bhuiyan, and Xuelian Lin. Safedrive: online driving anomaly detection from large-scale vehicle data. *IEEE Transactions on Industrial Informatics*, 13(4):2087–2096, 2017.

[31] Tung Kieu, Bin Yang, and Christian S Jensen. Outlier detection for multidimensional time series using deep neural networks. In *2018 19th IEEE international conference on mobile data management (MDM)*, pages 125–134. IEEE, 2018.

[32] Dejan Mitrovic. Reliable method for driving events recognition. *IEEE transactions on intelligent transportation systems*, 6(2):198–205, 2005.

[33] Ashutosh Kumar Choudhary and Piyush K Ingole. Smart phone based approach to monitor driving behavior and sharing of statistic. In *2014 fourth international conference on communication systems and network technologies*, pages 279–282. IEEE, 2014.

[34] Rosolino Vaiana, Teresa Iuele, Vittorio Astarita, Maria Vittoria Caruso, Antonio Tassitani, Claudio Zaffino, and Vincenzo Pasquale Giofrè. Driving behavior and traffic safety: an acceleration-based safety evaluation procedure for smartphones. *Modern Applied Science*, 8(1):88, 2014.

[35] Harsha Kumara Kalutarage, M Omar Al-Kadri, Madeline Cheah, and Garikayi Madzudzo. Context-aware anomaly detector for monitoring cyber attacks on automotive can bus. In *Proceedings of the 3rd ACM Computer Science in Cars Symposium*, pages 1–8, 2019.

[36] Bernhard Schölkopf, Robert C Williamson, Alex Smola, John Shawe-Taylor, and John Platt. Support vector method for novelty detection. *Advances in neural information processing systems*, 12, 1999.

[37] Edward W Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *biometrics*, 21:768–769, 1965.

[38] Charles A Bouman, Michael Shapiro, GW Cook, C Brian Atkins, and Hui Cheng. Cluster: An unsupervised algorithm for modeling gaussian mixtures, 1997.

[39] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

[40] Sebastian Pölsterl. scikit-survival: A library for time-to-event analysis built on top of scikit-learn. *J. Mach. Learn. Res.*, 21(212):1–6, 2020.

[41] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[42] Michael W Browne. Cross-validation methods. *Journal of mathematical psychology*, 44(1):108–132, 2000.

[43] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast bayesian optimization of machine learning hyperparameters on large datasets. In *Artificial intelligence and statistics*, pages 528–536. PMLR, 2017.

[44] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and David Sculley. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1487–1495, 2017.

[45] Claudio De Stefano, Carlo Sansone, and Mario Vento. To reject or not to reject: that is the question-an answer in case of neural classifiers. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(1):84–94, 2000.

[46] Odin Taylor and D Addison. Novelty detection using neural network technology. In *COMADEM 2000: 13 th International Congress on Condition Monitoring and Diagnostic Engineering Management*, pages 731–743, 2000.

[47] François Chollet et al. Keras. `https://keras.io`, 2015.

[48] Andreas Bulling, Ulf Blanke, and Bernt Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys (CSUR)*, 46(3):1–33, 2014.

[49] scikit-learn developers. scikit-learn documentation: Support vector machines. `https://scikit-learn.org/stable/modules/svm.html`, 01 2019. Accessed: 2023-03-13.

[50] Jean-Philippe Vert, Koji Tsuda, and Bernhard Schölkopf. A primer on kernel methods. *Kernel methods in computational biology*, 47:35–70, 2004.

[51] Jason Weston, Chris Watkins, et al. Support vector machines for multi-class pattern recognition. In *Esann*, volume 99, pages 219–224, 1999.

[52] Mahesh Pal. Multiclass approaches for support vector machine based land cover classification. *arXiv preprint arXiv:0802.2411*, 2008.

[53] Mohamed Aly. Survey on multiclass classification methods. *Neural Netw*, 19(1-9):2, 2005.

[54] D Powers. Evaluation: from precision, recall and f-factor to roc, informedness, markedness andamp; correlation. *J Mach Learn Technol*, 2:2229–3981, 2008.

[55] David MW Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*, 2020.

[56] Stan Aronoff et al. Classification accuracy: a user approach. *Photogrammetric Engineering and Remote Sensing*, 48(8):1299–1307, 1982.

[57] Clara Marina Martinez, Mira Heucke, Fei-Yue Wang, Bo Gao, and Dongpu Cao. Driving style recognition for intelligent vehicle control and advanced driver assistance: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 19(3):666–676, 2017.

[58] Empatica. Embraceplus. `https://www.empatica.com/en-eu/embraceplus/`, 2022. Accessed: 2022-11-01.

[59] Cedric Haase. Detecting dangerous driving situations from in-car sensor data via machine learning approaches. Bachelor's thesis, University of Siegen, 2019.

[60] Frédéric Li, Kimiaki Shirahama, Muhammad Adeel Nisar, Lukas Köping, and Marcin Grzegorzek. Comparison of feature learning methods for human activity recognition using wearable sensors. *Sensors*, 18(2):679, 2018.

[61] Tahera Anjuman, Shahnewaz Hasanat-E-Rabbi, Chowdhury Kawsar Arefin Siddiqui, Md Mazharul Hoque, et al. Road traffic accident: A leading cause of the global burden of public health injuries and fatalities. In *InProc. Int. Conf. Mech. Eng. Dhaka Bangladesh*, pages 29–31, 2020.

[62] World Health Organization et al. Global status report on road safety 2018: Summary. Technical report, World Health Organization, 2018.

[63] Luis M Bergasa, Daniel Almería, Javier Almazán, J Javier Yebes, and Roberto Arroyo. Drivesafe: An app for alerting inattentive drivers and scoring driving behaviors. In *2014 IEEE Intelligent Vehicles symposium proceedings*, pages 240–245. IEEE, 2014.

[64] Bruce R Donnelly, David Schabel, Alan J Blatt, and Arthur Carter. The automated collision notification system. In *Transportation Recording: 2000 and Beyond. International Symposium on Transportation RecordersNational Transportation Safety BoardInternational Transportation Safety Association*, 1999.

[65] Gys Albertus Marthinus Meiring and Hermanus Carel Myburgh. A review of intelligent driving style analysis systems and related artificial intelligence algorithms. *Sensors*, 15(12):30653–30682, 2015.

[66] Jorge Zaldivar, Carlos T Calafate, Juan Carlos Cano, and Pietro Manzoni. Providing accident detection in vehicular networks through obd-ii devices and android-based smartphones. In *2011 IEEE 36th Conference on Local Computer Networks*, pages 813–819. IEEE, 2011.

[67] Kristofer Kusano and Hampton C Gabler. Comparison and validation of injury risk classifiers for advanced automated crash notification systems. *Traffic injury prevention*, 15(sup1):S126–S133, 2014.

[68] Tetsuya Nishimoto, Kosuke Mukaigawa, Shigeru Tominaga, Nils Lubbe, Toru Kiuchi, Tomokazu Motomura, and Hisashi Matsumoto. Serious injury prediction algorithm based on large-scale data and under-triage control. *Accident Analysis & Prevention*, 98:266–276, 2017.

[69] Michelangelo-Santo Gulino, Leonardo Di Gangi, Alessio Sortino, and Dario Vangi. Injury risk assessment based on pre-crash variables: The role of closing velocity and impact eccentricity. *Accident Analysis & Prevention*, 150:105864, 2021.

[70] Carvaloo - thyssenkrupp automotive technology. `https://www.thyssenkrupp-automotive-technology.com/en/products-and-services/carvaloo`. Accessed: 2022-02-24.

[71] Transportation Research Board (TRB) of the National Academy of Sciences. The 2nd strategic highway research program naturalistic driving study dataset. 2013.

[72] Guillaume Leduc et al. Road traffic data: Collection methods and applications. *Working Papers on Energy, Transport and Climate Change*, 1(55):1–55, 2008.

[73] Unaiza Alvi, Muazzam A Khan Khattak, Balawal Shabir, Asad Waqar Malik, and Sher Ramzan Muhammad. A comprehensive study on iot based accident detection systems for smart vehicles. *IEEE Access*, 8:122480–122497, 2020.

[74] Haimd M Ali and Zainab S Alwan. *Car accident detection and notification system using smartphone*. LAP LAMBERT Academic Publishing Saarbrucken, 2017.

[75] Md Syedul Amin, Jubayer Jalil, and Mamun Bin Ibne Reaz. Accident detection and reporting system using gps, gprs and gsm technology. In *2012 International Conference on Informatics, Electronics & Vision (ICIEV)*, pages 640–643. IEEE, 2012.

[76] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[77] Karsten Jakobsen, Sabrine CH Mouritsen, and Kristian Torp. Evaluating eco-driving advice using gps/canbus data. In *Proceedings of the 21st ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 44–53, 2013.

[78] Brian C Tefft. Reducing risk and improving traffic safety: research on driver behavior and performance. *Institute of Transportation Engineers. ITE Journal*, 88(8):30–34, 2018.

[79] Jair Ferreira, Eduardo Carvalho, Bruno V Ferreira, Cleidson de Souza, Yoshihiko Suhara, Alex Pentland, and Gustavo Pessin. Driver behavior profiling: An investigation with different smartphone sensors and machine learning. *PLoS one*, 12(4):e0174959, 2017.

[80] Kawtar Zinebi, Nissrine Souissi, and Kawtar Tikito. Driver behavior analysis methods: Applications oriented study. In *Proceedings of the 3rd International Conference on Big Data, Cloud and Applications-BDCA*, pages 4–5, 2018.

[81] Samina Khalid, Tehmina Khalil, and Shamila Nasreen. A survey of feature selection and feature extraction techniques in machine learning. In *2014 science and information conference*, pages 372–378. IEEE, 2014.

[82] John-Syin Ang, Kok-Why Ng, and Fang-Fang Chua. Modeling time series data with deep learning: A review, analysis, evaluation and future trend. In *2020 8th International Conference on Information Technology and Multimedia (ICIMU)*, pages 32–37, 2020.

[83] Masataka Mori, Chiyomi Miyajima, Pongtep Angkititrakul, Takatsugu Hirayama, Yiyang Li, Norihide Kitaoka, and Kazuya Takeda. Measuring driver awareness based on correlation between gaze behavior and risks of surrounding vehicles. In *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pages 644–647. IEEE, 2012.

[84] Shaneel Bachoo, Anil Bhagwanjee, and Kaymarlin Govender. The influence of anger, impulsivity, sensation seeking and driver attitudes on risky driving behaviour among post-graduate university students in durban, south africa. *Accident Analysis & Prevention*, 55:67–76, 2013.

[85] Arash Jahangiri, Hesham A Rakha, and Thomas A Dingus. Adopting machine learning methods to predict red-light running violations. In *2015 IEEE 18th*

International Conference on Intelligent Transportation Systems, pages 650–655. IEEE, 2015.

[86] Eshed Ohn-Bar and Mohan M Trivedi. Beyond just keeping hands on the wheel: Towards visual interpretation of driver hand motion patterns. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1245–1250. IEEE, 2014.

[87] Chris Lee, Frank Saccomanno, and Bruce Hellinga. Analysis of crash precursors on instrumented freeways. *Transportation Research Record*, 1784(1):1–8, 2002.

[88] Omar Bagdadi. Assessing safety critical braking events in naturalistic driving studies. *Transportation research part F: traffic psychology and behaviour*, 16:117–126, 2013.

[89] Omar Bagdadi and András Várhelyi. Development of a method for detecting jerks in safety critical events. *Accident Analysis & Prevention*, 50:83–91, 2013.

[90] Charles Harlow and Yu Wang. Automated accident detection system. *Transportation research record*, 1746(1):90–93, 2001.

[91] Shunsuke Kamijo, Yasuyuki Matsushita, Katsushi Ikeuchi, and Masao Sakauchi. Traffic monitoring and accident detection at intersections. *IEEE transactions on Intelligent transportation systems*, 1(2):108–118, 2000.

[92] Jean Bacon, Andrei Iu Bejan, Alastair R Beresford, David Evans, Richard J Gibbens, and Ken Moody. Using real-time road traffic data to evaluate congestion. In *Dependable and Historic Computing*, pages 93–117. Springer, 2011.

[93] Jules White, Chris Thompson, Hamilton Turner, Brian Dougherty, and Douglas C Schmidt. Wreckwatch: Automatic traffic accident detection and notification with smartphones. *Mobile Networks and Applications*, 16(3):285–303, 2011.

[94] Li Chuan-zhi, Hu Ru-fu, and Hong-wu Ye. Method of freeway incident detection using wireless positioning. In *2008 IEEE International Conference on Automation and Logistics*, pages 2801–2804. IEEE, 2008.

[95] Jiuh-Biing Sheu. A sequential detection approach to real-time freeway incident detection and characterization. *European Journal of Operational Research*, 157(2):471–485, 2004.

[96] Adnan Bin Faiz, Ahmed Imteaj, and Mahfuzulhoq Chowdhury. Smart vehicle accident detection and alarming system using a smartphone. In *2015 International Conference on Computer and Information Engineering (ICCIE)*, pages 66–69. IEEE, 2015.

[97] Vasif Ahmed and Naresh P Jawarkar. Design of low cost versatile microcontroller based system using cell phone for accident detection and prevention.

In *2013 6th International Conference on Emerging Trends in Engineering and Technology*, pages 73–77. IEEE, 2013.

[98] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller Jensen. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM conference on embedded networked sensor systems*, pages 127–140, 2015.

[99] Murat Ozbayoglu, Gokhan Kucukayan, and Erdogan Dogdu. A real-time autonomous highway accident detection model based on big data processing and computational intelligence. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 1807–1813. IEEE, 2016.

[100] Bin Pan and Hao Wu. Urban traffic incident detection with mobile sensors based on svm. In *2017 XXXIInd General Assembly and Scientific Symposium of the International Union of Radio Science (URSI GASS)*, pages 1–4. IEEE, 2017.

[101] Sreyan Ghosh, Sherwin Joseph Sunny, and Rohan Roney. Accident detection using convolutional neural networks. In *2019 International Conference on Data Science and Communication (IconDSC)*, pages 1–6. IEEE, 2019.

[102] Osama A Osman, Mustafa Hajij, Peter R Bakhit, and Sherif Ishak. Prediction of near-crashes from observed vehicle kinematics using machine learning. *Transportation Research Record*, 2673(12):463–473, 2019.

[103] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[104] Diane J Cook and Narayanan C Krishnan. *Activity learning: discovering, recognizing, and predicting human behavior from sensor data*. John Wiley & Sons, 2015.

[105] Jiliang Tang, Salem Alelyani, and Huan Liu. Feature selection for classification: A review. *Data classification: Algorithms and applications*, page 37, 2014.

[106] Meng Lu. Embedded feature selection accounting for unknown data heterogeneity. *Expert Systems with Applications*, 119:350–361, 2019.

[107] Ryan J Urbanowicz, Melissa Meeker, William La Cava, Randal S Olson, and Jason H Moore. Relief-based feature selection: Introduction and review. *Journal of biomedical informatics*, 85:189–203, 2018.

[108] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[109] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[110] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

[111] Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*, 2014.

[112] John Cristian Borges Gamboa. Deep learning for time-series analysis. *arXiv preprint arXiv:1701.01887*, 2017.

[113] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[114] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[115] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[116] Oliver Kramer. K-nearest neighbors. In *Dimensionality reduction with unsupervised nearest neighbors*, pages 13–23. Springer, 2013.

[117] Anthony J Myles, Robert N Feudale, Yang Liu, Nathaniel A Woody, and Steven D Brown. An introduction to decision tree modeling. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 18(6):275–285, 2004.

[118] D Richard Cutler, Thomas C Edwards Jr, Karen H Beard, Adele Cutler, Kyle T Hess, Jacob Gibson, and Joshua J Lawler. Random forests for classification in ecology. *Ecology*, 88(11):2783–2792, 2007.

[119] Philip Gouverneur, Frédéric Li, Wacław M Adamczyk, Tibor M Szikszay, Kerstin Luedtke, and Marcin Grzegorzek. Comparison of feature extraction methods for physiological signals for heat-based pain recognition. *Sensors*, 21(14):4838, 2021.

[120] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiaoli Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Ijcai*, volume 15, pages 3995–4001. Buenos Aires, Argentina, 2015.

[121] Bendong Zhao, Huanzhang Lu, Shangfeng Chen, Junliang Liu, and Dongya Wu. Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, 28(1):162–169, 2017.

[122] Lamyaa Sadouk. Cnn approaches for time series classification. In *Time Series Analysis-Data, Methods, and Applications*, pages 1–23. IntechOpen, 2019.

[123] Frédéric Li, Kimiaki Shirahama, Muhammad Adeel Nisar, Xinyu Huang, and Marcin Grzegorzek. Deep transfer learning for time series data based on sensor modality classification. *Sensors*, 20(15):4271, 2020.

[124] Haoyue Liu, MengChu Zhou, and Qing Liu. An embedded feature selection method for imbalanced data classification. *IEEE/CAA Journal of Automatica Sinica*, 6(3):703–715, 2019.

[125] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[126] Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. Time-series generative adversarial networks. *Advances in neural information processing systems*, 32, 2019.

[127] Shayan Jawed, Josif Grabocka, and Lars Schmidt-Thieme. Self-supervised learning for semi-supervised time series classification. In *Advances in Knowledge Discovery and Data Mining: 24th Pacific-Asia Conference, PAKDD 2020, Singapore, May 11–14, 2020, Proceedings, Part I 24*, pages 499–511. Springer, 2020.

[128] Xiangyong Cao, Jing Yao, Zongben Xu, and Deyu Meng. Hyperspectral image classification with convolutional neural network and active learning. *IEEE Transactions on Geoscience and Remote Sensing*, 58(7):4604–4616, 2020.

[129] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963, 2019.

List of own publications

[1] Stenger, Roland, **Hawzhin Hozhabr Pour**, Jonas Teich, Andreas Hein, and Sebastian Fudickar. *Gait Parameters Estimation from Sensor-Belt IMU Data.* Authorea Preprints, 2024. Authorea.

[2] Ciortuz, Gabriela, **Hawzhin Hozhabr Pour**, and Sebastian JF Fudickar. *Evaluating Movement and Device-Specific DeepConvLSTM Performance in Wearable-Based Human Activity Recognition* BIOSTEC (2), 746–753, 2024.

[3] **Hozhabr Pour, Hawzhin** and Li, Frédéric and Wegmeth, Lukas and Trense, Christian and Doniec, Rafał and Grzegorzek, Marcin and Wismüller, Roland, *A machine learning framework for automated accident detection based on multimodal sensors in cars*, Sensors (MDPI) 10;22(10):3634, 2022.

[4] Doniec, Rafał and Piaseczna, Natalia and Li, Frédéric and Duraj, Konrad and Pour, **Hawzhin Hozhabr** and Grzegorzek, Marcin and Mocny-Pachońska, Katarzyna and Tkacz, Ewaryst, *Classification of roads and types of public roads using EOG smart glasses and an algorithm based on machine learning while driving a car*, Electronics (MDPI) 18;11(18):2960, 2022.

[5] **Hozhabr Pour, Hawzhin** and Wegmeth, Lukas and Kordes, Alexander and Grzegorzek, Marcin and Wismüller, Roland, *Feature extraction and classification of sensor signals in cars based on a modified codebook approach*, Progress in Computer Recognition Systems 11 (Springer), 184–194, 2020.

[6] Kordes, Alexander and Wurm, Sebastian and **Hozhabrpour, Hawzhin** and Wismüller, Roland, *Automatic Fault Detection using Cause and Effect Rules for In-vehicle Networks.*, VEHITS 537–544, 2018.

List of abbreviations

Acc ..... Accuracy
ACN ..... Automatic Crash Notification
ADAS ..... Advanced Driver Assistance Systems
AE ..... Autoencoder
AF1 ..... Average F1-score
AFS ..... Automated Feature Selection
ANN ..... Artificial Neural Network
ARC ..... Activity Recognition Chain
AVA ..... All-Versus-All
CAN ..... Controller Area Network
CB ..... Codebook approach
CNN ..... Convolutional Neural Network
DAE ..... Denoising Autoencoder
DNN ..... Deep Neural Network
DTW ..... Dynamic Time Warping
IoT ..... Internet of Things
GAN ..... Generative Adversarial Network
GMM ..... Gaussian Mixture Model
HCF ..... Handcrafted Features
HFS ..... Handcrafted Feature Selection
kNN ..... k-Nearest-Neighbors
LSTM ..... Long-Short-Term Memory
mDNN ..... multichannel Deep Neural Network
ML ..... Machine Learning
MLP ..... Multi-Layer Perceptron
MSPRT ..... Modified Sequential Probability Ratio Test
NDS ..... Naturalistic Driving Study
OBU ..... On-Board Unit
OVA ..... One-Versus-All
PCA ..... Principal Component Analysis
pp ..... predicted positives
PRC ..... Pain Recognition Chain
RBF ..... Radial Basis Function
RELU ..... Rectified Linear Unit
RFE ..... Recursive Feature Elimination
RNN ..... Recurrent Neural Networks
RPM ..... Revolutions Per Minute

RSUs ..... Roadside Units
RTMS ..... Real-Time Monitoring System
sDNN ..... single channel Deep Neural Network
SG ..... State Graph
SHRP2 ..... the second Strategic Highway Research Program
SVM ..... Support Vector Machine
tp ..... true positives
TPA ..... True Positive Accuracy
TPR ..... True Positive Rate
VTTI ..... Virginia Tech Transportation Institute
WHO ..... World Health Organization

List of Tables

List of Figures