Deep Learning-Based Driver Behavior Detection on Simulated and Real Data

by

Zehra Camlica

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2023

© Zehra Camlica 2023

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Dr. Giulio Bianchi Piccinini Department of the Built Environment,Aalborg University
Supervisor(s):	Mark Crowley Professor, Dept. of ECE, University of Waterloo
Internal Member:	Sebastian Fischmeister Professor, Dept. of ECE, University of Waterloo Fakhri Karray Professor, Dept. of ECE, University of Waterloo
Internal-External Member:	Baris Fidan Professor, Dept. of MME, University of Waterloo

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners. I understand that my thesis may be made electronically available to the public.

Abstract

Driver behaviour has a significant influence on vehicle accidents. Measuring and providing feedback on driver behaviour can provide significant benefits for understanding and improving road safety. Mobile phones can be leveraged for the detection of driver actions and characteristics from the broadest population of drivers. Mobile phones also offer easy accessibility for cost-effective and reliable information with the built-in sensors available on them, such as the Global Positional System (GPS) and Inertial Measurement Unit (IMU). However, when it comes to a larger scale, obtaining labelled data from these mobile devices is still far from optimal for low-cost and reliable applications due to noise and missing data. In this study, data obtained from mobile phone sensors is simulated as a time series dataset using a traffic simulator and a robotic simulator. Then, the dataset is used with deep learning methods to classify both manoeuvres and driver behaviours, focusing specifically on aggressive driver behaviour. We propose a novel method using two Convolutional Neural Networks Convolutional Neural Networks (CNN) working in parallel to classify driver behaviours while classifying manoeuvres (i.e., aggressive right lane-change). We claim that the Parallel Convolutional Neural Network (PCNN) not only speeds up training time but also increases performance since having information about the manoeuvre helps improve behaviour classification performance. To validate this, first, a single task CNN for manoeuvre classification and a single task CNN aggressive/non-aggressive behaviour classifiers were built separately. The utility of the classifiers was demonstrated on a large simulated dataset created using the Sumo and Webots simulators. Subsequently, the PCNN classifier has been trained and validated on the big simulated dataset and a small driven dataset. We have also collected a dataset driven on real road by using GPS and IMU sensors and the PCNN model has been tested on this real dataset to investigate whether a classifier trained on simulated data can generalize to real data. In addition to this method, we propose a method of Spatial CNN with Attention (SCNN-A) layer to apply to our time series data with extracting more high-level features from spatial data for classification purposes.

Acknowledgements

First of all, I would like to express my sincere gratitude to my advisor, Prof. Mark Crowley for supervising my studies and encouraging me towards pursuing a graduate degree. Without his continuous support, patience and encouragement, the studies presented here could not possibly have been accomplished. I have learned a lot from him through our discussions. I owe a debt of gratitude to my committee members Profs. Baris Fidan, Sebastian Fischmeister, Fakhri Karray and Dr. Giulio Bianchi Piccinini for reading my thesis. I appreciate the considerable amount of time and effort they have invested. I extend my deepest heartfelt thanks to my family Yusuf Asaf Camlica, Zeynep Camlica, Ahmet Camlica, Kadriye Aydin, Ahmet Aydin, Ali S. Camlica, Yunus Camlica, Zohre Camlica, Munteha Pac, Ayse Ozaydin, Osman M. Aydin and Abdullah Camlica for their faithful support and confidence in me. I would always be indebted to them. Thank you to Ayca Diriksoy, Ayse Erenay , Ayse Kuspinar, Derya Kara, Feyza Zeytinoglu, Gizem S. Nemutlu Dalgic, Ilknur Umay, Laura McCrackin, Nada Gohider, Nezahat Ider, Sinem Orbay, Zeynep Korkmaz and Zeynep Sertbas for their endless friendship and support, and also thanks to my moral older sisters Betul Kara, Hulya Biyikli and Zennure Abla for inspirational advice that I will never ever forget throughout my life. Thank you all to my dearest friends.

Finally, I would like to thank my creator, God, for every time I had in my life with feeling as near me.

Dedication

To my God Almighty.

Table of Contents

Examining Committee Membership ii

Author's Declaration iii

Abstract iv

Acknowledgements v

Dedication vi

List of Figures x

List of Tables xii

List of Abbrevations xiv

1 Introduction 1

1.1 Driving Behaviours 2

1.1.1 Driving Manoeuvres 2

1.1.2 Driving Characteristics 3

1.2 Applications Assessing Driver Behaviours 6

1.2.1 Driver Performance Assessment 6

1.2.2 Intelligent Vehicle Systems and Autonomous Vehicles 7

1.3 Motivation 8
1.4 Objective of This Research 9
1.5 Organization of The Thesis 11

2 Literature Review and Background **12**
2.1 Literature Review 12
2.2 Artificial Neural Networks 14
2.2.1 Activation Functions 15
2.2.2 Loss Functions 17
2.2.3 Optimization 18
2.2.4 Back Propagation 20
2.2.5 Network Architectures 20
2.3 Convolutional Neural Networks(CNN) 24
2.3.1 Convolution Layer 24
2.3.2 Pooling Operation 25
2.3.3 Statistical Analysis 25
2.4 Conclusion 26

3 Stacked Sparse Auto-encoder Method to Select Driver Behaviour Features **28**
3.1 Introduction 28
3.1.1 Windowing 29
3.1.2 Sparse Autoencoders 30
3.2 Experimental Set-up and Dataset 32
3.2.1 SSAE Implementation 33
3.3 Classification Results 37
3.4 Conclusion 41

4 Parallel CNN Classification for Aggressive Driver Behaviour **42**
4.1 Introduction . . . 42
4.2 Tools . . . 43
4.2.1 SUMO Simulation Tool . . . 44
4.2.2 Webots . . . 47
4.2.3 Keras and Tensorflow . . . 48
4.3 Experimental Setup and Dataset . . . 49
4.3.1 Simulated Dataset . . . 50
4.3.2 Real-World Driving Behaviour Learning Dataset (DBL) . . . 54
4.3.3 Convolutional Neural Network Implementation . . . 57
4.3.4 Parallel Convolutional Neural Network Implementation . . . 58
4.3.5 Results . . . 60
4.4 Conclusion . . . 62

5 Spatial PCNN Classification for Aggressive Driver Behavior with Attention layer **71**
5.1 Introduction . . . 71
5.2 Tools . . . 72
5.2.1 Spatial Convolutional Neural Network (SCNN) . . . 72
5.2.2 Attention . . . 74
5.3 Parallel Spatial Convolutional Neural Network implementation with Attention 77
5.4 Experimental Analysis on the Real-world Driven DBL Dataset . . . 78
5.5 Results . . . 79
5.6 Conclusion . . . 85

6 Conclusion **86**
6.1 Summary and Conclusion . . . 86
6.2 Thesis Contributions and Impact of the Study . . . 87
6.3 Future works . . . 89

References **90**

List of Figures

1.1 General system design 10

2.1 An Overview of Fundamental Activation Functions. 15
2.2 An example of Sparse AE 23

3.1 The Flow Diagram of the Method 29
3.2 The Windowing Technique Application for Pre-processing 30
3.3 This Illustrates the Labelled Dataset Information 33
3.4 The 3D Illustration of Features for Window Size 10 34
3.5 The 3D Illustration of Features for Window Size 50 35
3.6 The 3D Illustration of Features for Window Size 100 36
3.7 The 3D Illustration of Features which are Selected by PCA 37

4.1 General System Architecture for Driver Behaviour Analysis 44
4.2 An Illustration of SUMO-gui 45
4.3 An illustration of Webots 48
4.4 A Rotated Left Turn Time Sequence Sample Conversion Steps to Image . . 52
4.6 Vehicle Angle of Right Turn Sequences and Aggressive Right Turn Sequences from the Dataset 53
4.10 Automated Data Labelling by Thresholding and Scoring Flowchart 56
4.5 Zero Centered Vehicle X and Y Position Data Samples for 300 Time Window from the Dataset 65

4.7 Two Different Participants' Route-speed Plot 66
4.8 The Route that Is Used for Participant Drives 67
4.9 Manual Labelling Tool 67
4.11 2D-CNN Architecture 67
4.12 Parallel CNN Architecture for Two Different Tasks, Driving Manoeuvres and Driver Behaviors 68
4.13 PCNN Classification on **Rotated Time Series of Simulated Data** Accuracy and Loss vs Epochs plotting: 69
4.14 PCNN Classification on **Real Driven Dataset** Accuracy and Loss vs Epochs plotting 70

5.1 Spatial-CNN Architecture for Image Classification in [1] 73
5.2 Attention-CNN Architecture for Image Classification in [2] 75
5.3 SPCNN with Attention Architecture for Driver Behaviour Classification . . 77
5.4 Orange3 Visualization Tool and Illustration of Latitude and Longitude Features with Turn Labels to Check Overlapping Time 80
5.5 Two Different Participants' Route-speed Plot: 81
5.6 SPCNN-a Classification on DBL Dataset (for Four Manoeuvre Classes) Accuracy and Loss vs Epochs plotting 83
5.7 SPCNN-A Classification on DBL Dataset (for 7 Manoeuvre Classes) Accuracy and Loss vs Epochs Plotting 84

List of Tables

3.1 Accuracy and F1 Score . . . 38
3.2 Labels . . . 38
3.3 Confusion Matrix for Window Size 10 (in %) . . . 39
3.4 Confusion Matrix for Window Size 50 (in %) . . . 39
3.5 Confusion Matrix for Window Size 100 (in %) . . . 40
3.6 Confusion Matrix for PCA (in %) . . . 40

4.1 Type of Classes . . . 49
4.2 SUMO Parameters for Aggressive and Normal Driver Behaviours . . . 51
4.3 PCNN Layers Windows and Output Dimensions for Plain or Rotated Time Series Data Input . . . 59
4.4 PCNN Layers Windows and Output Dimensions for Converted Image Data Input: . . . 59
4.5 Aggressive Driver Behaviours Determination Rules and Thresholds . . . 60
4.6 Accuracy Results for Different Tasks Using just the Single Task CNN on Time Series Datasets. . . . 61
4.7 General Accuracy Results of Classification for 8 Classes Together with Various Models Applied on Whole **Plain Simulated Time Series Dataset** . 63
4.8 PCNN Application Results Comparison Table on the Different Type of Simulated data . . . 63

5.1 List of Manoeuvre and Aggressive Behaviour Labels Used SPCNN-Att Classification Task . . . 79

5.2 Training and Test Accuracies for the PCNN, SPCNN, and SPCNN-Att Methods on Two Tasks 82
5.3 F1 Scores for Different Methods and Tasks 85

List of Abbreviations

GPS Global Positional System

IMU Inertial Measurement Unit

CNN Convolutional Neural Networks

PCNN Parallel Convolutional Neural Network

SCNN-A Spatial CNN with Attention

PSCNN-A Parallel Spatial CNN with Attention

AI Artificial Intelligence

ADAS Advanced Driver Assistance Systems

DAS Driver Assistance Systems

IVS Intelligent Vehicle Systems

ANNs Artificial Neural Networks

HMMs Hidden Markov Models

GMMs Gaussian Mixture Models

SVMs Support Vector Machines

LSTM Long Short-Term Memory

GCNs Graph Convolutional Networks

DTW Dynamic Time Warping

SGD Stochastic Gradient Descent

GD Gradient Descent

MLP Multilayer Perceptron

RNN Recurrent Neural Network

AE Autoencoders

DBL Driver Behaviour Learning

SCNN Spatial Convolutional Neural Network

ReLu Rectified Linear Unit

SAE Sparse Autoencoders

SSAE Stacked Sparse AutoEncoders

PCA Principal Component Analysis

KNN K Nearest Neighbour

Chapter 1

Introduction

There are three main contributing factors to road traffic accidents, human error, vehicle failure, and road conditions, where the first factor is the most influential [3]. Vehicle accidents are highly influenced by driver behaviour, and understanding driver behaviour is key to improving road safety. Furthermore, driving behaviour can be positively influenced when a driver is monitored and behaviours are recorded [4]. Therefore, methods for the detection and monitoring of driving behaviours are of great interest. There are many different driver behaviour detection systems that try to ensure both driver safety and compliance with driving regulations [5]. Traditional methods such as statistical analysis and data mining have been used for analyzing driver behaviour, but recent advancements in Artificial Intelligence (AI) and machine learning have opened up new possibilities for a more comprehensive analysis of driver behaviour. Advanced Driver Assistance Systems (ADAS) have utilized classification and pattern recognition approaches [6].

This thesis describes three different methods for driving behaviour analysis, which are Stacked Sparse Auto Encoders as a feature extraction method and the parallel Convolutional Neural Networks and Spatial Convolutional Networks with Attention layer as classification methods. This chapter gives a description of driving behaviours. In addition, the historical development of detecting driver behaviours is summarized. At the end of this chapter, the motivation and the objective of my research are briefly described. But more broadly, I will describe my thesis research in the following chapters.

1.1 Driving Behaviours

Driving behaviour is a generic term including different driving manoeuvres and driving mannerisms consisting of many variables. This concept is crucial in understanding and evaluating how a person operates a vehicle on the road. Driver behaviour systems can be analyzed into two complementary sub-problems: detecting driver actions and analyzing driving characteristics.

Driver manoeuvres are specific actions which a driver makes while operating a vehicle. For instance, these actions can include turns, lane changes, stops, as well as accelerating and decelerating events. These actions can greatly influence overall road safety.

Driving characteristics depends on how the drivers perform these actions, these characteristics can be derived from patterns and frequency of driving actions. The most common driving characteristics defined in the area are aggressive driving, distracted driving, and fatigued driving. Analyzing driver actions and driving characteristics are two interconnected areas. By focusing on these areas, solutions can be found to promote safer roads and it will help to model human driver behaviour.

1.1.1 Driving Manoeuvres

Manoeuvres are parts of a trip consisting of characteristics of driving patterns. Driver actions, which can also be referred to as driving manoeuvres, form the basis for understanding individual driving behaviour. Although there are no standardized categories for manoeuvres in existing literature, several commonly recognized types include [7]:

- **Stop/Acceleration/Deceleration:** These manoeuvres include the vehicle's speed changes, i.e. stopping, accelerating, and decelerating.
- **Left/Right Lane Change:** These manoeuvres refer to drivers moving their vehicles between lanes for various reasons i.e. overtaking.
- **Left/Right Intersection Turns:** These manoeuvres are making proper turns at intersections with obeying traffic signals and rules.
- **Road Curvature:** This manoeuvre means navigating curved roads, where drivers must adjust car speed and steering angle to maintain control of the vehicle.

- **Roundabouts:** These manoeuvres include circular intersections in order to manage traffic flow. Drivers are required to give priority to vehicles already in the circle before entering, then they need to navigate and exit the circle.
- **Highway Ramps:** These manoeuvres are entry and exit points on highways and drivers need to merge into or out of the highway by doing proper direction and speed adjustments.

To identify and analyze these manoeuvres types, vehicle kinematics such as position, speed, acceleration, and steering angle, should be taken into account. For instance, stop, acceleration, and deceleration manoeuvres can be determined by examining speed [8] and longitudinal acceleration data [9]. Lane changes and intersection turns are also inferred from angular velocity and lateral acceleration patterns[10, 11].

1.1.2 Driving Characteristics

Various driving characteristics have been identified in the literature [12], [13], reflecting the diverse behaviours exhibited by drivers on the road. Some of these characteristics include speeding, distracted driving, fatigued driving, inattentive driving, aggressive driving, and drunk driving. This part will cover two contrasting driving behaviours: normal (or safe) driving and aggressive driving.

Normal or safe driving is characterized by responsible and cautious behaviour behind the wheel. Drivers who exhibit this behaviour adhere to traffic regulations, maintain a safe following distance, and are attentive to their surroundings. They adjust their driving style according to road and weather conditions, ensuring the safety of themselves and others on the road. Safe drivers also exhibit patience and courtesy, yielding to pedestrians and giving way to other vehicles when necessary [14].

On the other hand, aggressive driving consists of all driving actions which pose a danger for road safety. These aggressive behaviour characteristics are done by impatience, hostility, and a disregard for the safety and well-being of other road users [15]. Some actions that aggressive drivers may engage in are excessive speeding, tailgating, frequent and unsafe lane changes, running red lights, and displaying rude gestures or verbal aggression toward others [16].

Normal Driving Characteristics

Driver behaviour applications aim to detect out-of-the-ordinary behaviours, as noted by Mohamad et al. [17]. For this reason, defining normal driving characteristics is crucial for determining driver behaviour types. Different styles of driving can be derived by calculating deviations according to these normal driving characteristics. Some non-ordinary driving behaviour types include aggressive, inattentive, and drunk driving. Many studies have used different terms for ordinary driving styles, such as normal driving, safe driving, typical driving, or non-aggressive driving.

Some research, such as the study by Aljaafreh et al. [18], has categorized data gradually with different levels of driving behaviours (such as below normal, normal, aggressive, and very aggressive) by measuring how far away the data is from the norm. Another study aimed at detecting abnormal driving events defined normal driving parameters as speed and lateral position change [17]. To differentiate driving styles, some thresholds are determined using abnormalities from the standard, as seen in the study by Dorr et al.[19].

It is important to note that there may be numerous variations of normal driving styles since driving behaviour consists of many variables. For this reason, some studies, i.e [20], have opted to focus on training individual driving history instead of using general rules or thresholds when determining driving styles. By considering individual driving history, researchers can develop personalized models that better capture each driver's behaviour characteristics.

In summary, defining normal driving characteristics is a vital step in understanding and evaluating driver behaviours. By considering individual driving histories, established norms from various studies, and personalized thresholds, researchers can more effectively identify deviations and non-ordinary behaviours that may pose risks to road safety.

Aggressive Driving Characteristics

Aggressive driving encompasses a range of unsafe behaviours, including actions such as speeding, tailgating, abrupt acceleration, excessive braking, and hasty lateral position changes [18]. A considerable number of fatal accidents involve at least one element of aggressive driving [9, 21], underscoring the impact of these hazardous behaviours on road safety.

Various factors can lead to aggressive driving, such as impatience, frustration, hostility, or a desire to reduce travel time [22]. Furthermore, Houston et al.'s psychological study

[16] suggests that aggressive driving not only endangers public safety but also represents a set of maladaptive social behaviours.

In order to tackle the aggressive driving behaviour problem, its characteristics should be studied broadly. For this reason, many ADAS are developed to give feedback and provide effective solutions about any behaviour types of drivers to the users and the provider.

Components of Aggressive Driving and Examples

1. **Speeding and Racing:** A driver racing over the speed limit on Highway X.
2. **Tailgating:** A motorist following cars too closely on Main Street.
3. **Frequent and Unsafe Lane Changes:** A vehicle weaving between lanes without using indicators.
4. **Disregarding Traffic Controls:** Ignoring a red light at an intersection.
5. **Impaired Driving:** Driving under the influence on any road.

Measurement and Determination Methodologies

1. **Self-Reported Measures:** Surveys like the Driving Anger Scale offer insights into behaviors and attitudes related to aggressive driving [23].
2. **Observational and Naturalistic Studies:** These studies provide real-world insights into driving behaviors and interactions [24].
3. **Simulator-Based Research:** This allows the controlled study of specific variables contributing to aggressive driving [25].
4. **Analysis of Official Records:** Analyzing traffic violation and accident records offers objective data [26].

Rules and Thresholds for Determination

- Speeding: Considered aggressive when driving over 10 mph above the speed limit.
- Following Distance: A gap of less than two seconds is indicative of aggressive driving.

- Lane Changes: Frequent lane changes without signalling signify aggression.
- Traffic Violation Accumulation: Multiple traffic violations within a specified period are indicative of aggressive driving.

Psychological and Societal Context

Aggressive driving is influenced by individual psychological states, societal norms, road conditions, and legislative frameworks. Congested urban areas might exhibit higher instances of aggressive driving due to increased stress and frustration among drivers. Aggressive driving leads to increased accidents, road rage incidents, and traffic disruptions. Strategies such as public awareness campaigns, stringent legislation, and enhanced traffic enforcement are employed to mitigate its impact.

1.2 Applications Assessing Driver Behaviours

Applications of driver behaviour and driving style analysis are categorized according to their purposes. An investigation based on these categories enables us to identify recent developments and omissions in that area. Some of the categories covered in this chapter are driver performance assessment, intelligent vehicle systems, and autonomous vehicles.

1.2.1 Driver Performance Assessment

Driving is an activity including complex processes since it consists of several components such as; driver, vehicle, and environment. In order to assess driver performance, these components and their relationships should be investigated as key factors [27]. Accordingly, driver age, experience, physical, emotional, and psychological state, vehicle specifications, and environmental conditions directly affect driving abilities. Driver performance is evaluated according to these factors by creating applications such as driver assistance and insurance applications.

Driver Assistance

Driver Assistance Systems (DAS) have gained more interest with the development of intelligent vehicle systems [19, 28]. The most important reason for this interest is road safety.

The main purpose of DAS is to bring about awareness to the driver of potential risky actions [9, 29]. Analyzing driving behaviour in real-time and giving feedback to the driver affects the driver's attention and increases road safety [30, 31]. On the other hand, DAS provides information about surrounding conditions to keep the driver alert to preventable road accidents [32]. ADAS are improved form of DAS for driver safety. Emergency braking systems, lane-keeping assistance systems, and collision warning systems are some common examples of ADAS [30, 33, 34, 35].

Insurance Applications

Statistical data has been used to evaluate drivers and to calculate the expected cost of future insurance claims. This data consists of significant factors such as the vehicle's type, value, characteristics, and driver's profile (age, gender, driver experience, etc.) [36]. For example, the study in [37] investigates the calculation of a score for an individual driver by detecting risky driving actions. In order to retrieve GPS information and to use this information for an estimation of risk score, insurance companies provide in-vehicle units to users. Another in-vehicle system is the DriveDiagnostics system. It analyzes driver behaviours by calculating risk and provides relevant feedback while on the road [38]. Constantly monitoring with in-vehicle sensors can be disturbing or uncomfortable for some drivers. Therefore, smartphone sensing platforms have been studied in [7, 39]. The smartphone-based systems can also serve as ADAS or support tools [33], and can provide feedback to a driver to reduce risky events in order to get premium insurance. For instance, Pay-As-You-Drive calculates insurance plans based on the driver's risk profile using a smartphone system [40]. The insurance plan starts with an initial plan and is recalculated after a predefined evaluation period using accelerometer, GPS, or onboard diagnostics data. Furthermore, MobiDriveScore not only calculates the drivers' profile but also enables a customer to monitor their personal driving history and score by using a smartphone [41].

1.2.2 Intelligent Vehicle Systems and Autonomous Vehicles

Intelligent Vehicle Systems (IVS) have been equipped with sensors to collect information from the vehicle, driver, and surrounding vehicles in order to evaluate them for feedback to the driver [42]. Finding lane-changing paths, investigating yaw angle and steering wheel relationship using inertial measurements, and estimating vehicle orientation using GPS are some common examples of information evaluated by IVS [43]. Driver behaviour modelling and driver style classification play important roles in autonomous vehicle research [12].

Autonomous vehicles should cooperate with the surroundings of vehicles for safe travel. It is inevitable that an autonomous vehicle will encounter human-driven vehicles, and therefore the prediction of future driver behaviour must provide sufficient information for the decision-making process to decrease the probability of risky events [44, 45]. IVS developments have a direct impact on driver behaviour analysis applications.

Recent advancements in IVS and autonomous vehicles include the use of artificial intelligence and machine learning techniques for improved driver behaviour analysis and prediction [13]. Some of these techniques involve deep learning models for detecting and classifying aggressive driving behaviours [36] and reinforcement learning algorithms for enabling autonomous vehicles to adapt their driving strategies based on the observed behaviour of surrounding human-driven vehicles [46]. Furthermore, recent research has also focused on the development of cooperative driving strategies for a mixed traffic environment, where autonomous and human-driven vehicles need to efficiently and safely coexist [47].

1.3 Motivation

Driving-related accidents have become a significant concern worldwide, with the majority of such accidents attributed to driver errors. Studies have shown that over 90 percent of accidents are caused by driver errors [48, 49]. This risk has been exacerbated by an increase in the number of licensed drivers over the past two decades [50, 51]. To mitigate this increasing risk, the development of advanced vehicle safety systems is essential, and these systems hinge on the accurate analysis of driver behaviour [35, 28].

Aggressive driving events such as speeding, running red lights, tailgating, weaving in and out of traffic, and failing to yield the right of way have been identified as significant contributors to road accidents [52]. Interestingly, the majority of drivers who exhibit aggressive driving behaviours are often unaware that they have committed any violations [4, 53]. Studies have also shown that driver behaviour can be positively influenced when drivers are monitored, and their behaviours are recorded [4, 31]. Therefore, monitoring and recording driver behaviour can lead to positive changes, making roads safer.

To detect driver actions and characteristics, there are different data sources: in-vehicle systems and smartphones. In-vehicle systems, such as GPS, IMU, and video cameras, have traditionally been used in fleet management systems and public transportation vehicles [54, 43]. However, with the rapid advancement of information and communication technology, smartphones have emerged as a promising tool for collecting real-time spatio-temporal information [55, 39].

Despite the potential challenges such as handling noisy sensor data and dealing with missing data in real-time series [56, 31], smartphones offer a cost-effective and efficient solution for collecting and analyzing a wealth of information about vehicle dynamics and contextual settings. As such, the use of smartphones for deriving driver behaviour profiles becomes an attractive alternative to in-vehicle data recorders [39, 57].

The problem of collecting, storing, and processing data while preserving privacy and security is a significant computational challenge. Techniques like anonymization, encryption, differential privacy, and secure multi-party computation can be considered, but each has its own computational overheads and implementation difficulties. Aggressive driver behaviour profiles would require a high volume of data to be processed and analyzed. This involves designing and implementing efficient algorithms and models to interpret the data and extract meaningful insights. If the system is expected to provide real-time feedback or immediate identification of aggressive behaviour, it poses another computational challenge. Real-time processing requires highly optimized algorithms and models and often a trade-off between accuracy and speed. In conclusion, the analysis of different driver behaviour profiles is paramount for improving traffic safety. This leads to our research question: How can we derive aggressive driver behaviour profiles from cost-efficient data while keeping drivers' privacy and security? This question lays the groundwork for our research, aiming to create a safer and more efficient transportation system [35, 28].

1.4 Objective of This Research

The assessment and classification of driver behaviour, particularly the identification of aggressive driving, is a critical area of research in road safety [39, 56]. Many different driver behaviour characteristics can be defined by using multiple measurements. A smartphone device can accurately label the types of manoeuvres and classify whether the driver is aggressive or not. In this research, our ultimate aim is to classify aggressive and non-aggressive driving behaviours by applying efficient, low-cost and robust techniques to basic sensor data (i.e. speed, accelerometer, gyroscope or GPS data). In order to achieve that, the first part of our approach involves the investigation of specific driver manoeuvres using real driving data collected via smartphones. A deep data feature reduction method is used to analyze and classify driving manoeuvres. The inherent complexity of this data necessitates the use of a deep data feature reduction method, which allows us to streamline the analysis and classification of different manoeuvres. After that, parallel working CNNs have been utilized to efficiently identify different driver behaviours by using both simulated vehicle data. Then the method is applied on a real-driven dataset. This driver behaviour

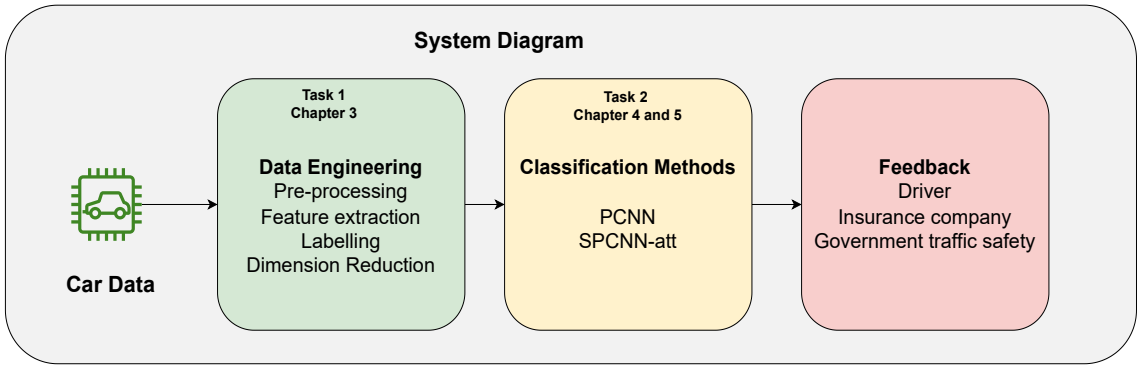


Figure 1.1: General system design

characteristics application predicts driver behaviours with reasonable accuracy, the method efficiently contributes to the purpose of this study.

There is a huge difference between aggressive driving with and without perception. Finding important parameters that show observation of aggressive behaviours makes individual acts and prevents accidents. However, it is not a simple pattern recognition problem with including complex relationships between features and parameters. Using deep learning techniques is beneficial in identifying affecting factors on aggressive driving behaviour. These factors can then be used to determine some countermeasures to reduce aggressive driving behaviour-related incidents and crashes.

In this research, I present three distinct methods for tackling the task at hand and evaluate their accuracy in detecting manoeuvres and classifying driver behaviour. Our overarching **aim is to identify aggressive driving behaviours and general driving manoeuvres by applying low-cost, efficient and robust methods using less intrusive sensors' data while keeping the privacy and security of drivers**. My research makes a significant contribution with proposing a machine-learning algorithm and evaluating combinations of machine-learning algorithms on both smartphone-based data, large simulated dataset, and real-driven dataset to detect aggressive driving events. In addition, I broadened the theoretical knowledge base by introducing a unique dataset combining different simulator tools and a real-driven dataset while performing various manoeuvres.

1.5 Organization of The Thesis

Chapter 2 will give an overview of different methods, algorithms and other important concepts used throughout the study, and a background of driving behaviour analysis. The theory behind the different algorithms will be presented along with general suggestions on how and when to use them. Chapter 3 presents a deep learning method which has already been published as a conference paper to identify a low-dimensional feature set for driver action recognition [58]. I will propose a different windowing method to generate the input data to the autoencoders, which investigates the influence of the network depth. In Chapter 4, a new deep learning method designed for a multi-tasking approach is demonstrated. I introduce a novel CNN Architecture for driver behaviour classification. It is called PCNN which refers to its dual structure of shared and parallel training [59]. A vehicle dataset is simulated using SUMO traffic simulator and Webots robotic simulator for aggressive driver behaviour detection. I demonstrate initial results with this basic model. Then in Chapter 5, I explore the number of improvements on top of PCNN such as new spatial CNN and attention layers. Finally, I will conclude the thesis's general contribution of the area in Chapter 6.

Chapter 2

Literature Review and Background

2.1 Literature Review

The most common methods and their applications in driver behaviours are covered in this chapter. Various methods have been employed to analyze and understand driver behaviours, including Artificial Neural Networks (ANNs) [60], Hidden Markov Models (HMMs) [44, 61], Fuzzy Logic [62, 18, 63, 19], Dynamic Time Warping (DTW) [9], Gaussian Mixture Models (GMMs) [64, 65, 43], and Support Vector Machines (SVMs) [66, 67, 68, 69].

Artificial Neural Networks (ANNs): ANNs are computational models that consist of interconnected processing elements known as neurons which work together to process and transmit information. Artificial Neural Networks have gained more interest recently [60]. Cheng et al. (2018) implemented an ANN to classify aggressive, normal, and calm driver behaviours [60]. [70, 62, 71] are some examples of Artificial Neural Networks applications based on computer vision techniques in drowsiness detection, distraction detection and steering behaviour prediction respectively. In an earlier study, Daza et al. (2011) employed an ANN for drowsiness detection [70]. Okuda et al. (2013) highlighted the challenges in estimating drivers' mental and physical states and collecting adequate information, emphasizing the increasing complexity of driver assistance applications. In a recent study, Yang et al. (2021) proposed the use of Long Short-Term Memory (LSTM) networks to model and predict driver lane-changing behaviours [72]. Another recent paper by Zhang et al. (2022) introduced a hybrid model combining Convolutional Neural Networks (CNNs) and LSTM networks for driver drowsiness detection [73]. These recent developments highlight the effectiveness of ANNs in capturing complex driver behaviours.

Hidden Markov Models (HMMs): Hidden Markov Models have been widely applied in driver behaviour analysis. HMMs are stochastic processes that can learn structure from observation sequences, making them applicable for driver behaviour analysis.Takeda et al. (2011) employed HMMs for driver distraction detection [64], while Gadepally et al. (2011) and Gadepally et al. (2014) used HMMs to estimate driving behaviour [44, 61]. In a more recent study, Yang et al. (2020) proposed a driver behaviour analysis framework based on Hidden Semi-Markov Models (HSMMs) to capture the temporal dynamics of driving events [74]. These advancements in HMM-based approaches enhance the accuracy and reliability of driver behaviour analysis. Kuge et al. (2000) utilized HMMs to characterize and detect driving manoeuvres [75].

Fuzzy Logic: Fuzzy logic-based methods have shown promise in driver behaviour analysis as they allow representing logic as a degree of truth. Tango et al. implemented fuzzy logic methods to identify driver fatigue and distraction [62]. Castignani et al. applied fuzzy logic to score driver behaviours [37], while Aljaafreh et al., Choudhary et al., and Dorr et al. used fuzzy logic to recognize driving styles [18, 63, 19]. In a recent paper, Zhao et al. proposed a novel fuzzy logic-based approach for driving style recognition using multiple features, including acceleration, speed, and steering wheel angle [76]. These recent developments highlight the effectiveness of fuzzy logic in capturing nuanced driver behaviours.

Dynamic Time Warping (DTW): Dynamic Time Warping, a pattern recognition method, has been used for driver behaviour analysis, particularly in classifying driver risk profiles and providing driver assistance. Johnson and Triverdi developed a smartphone driver analysis application called MIROAD, which utilized DTW to detect aggressive and non-aggressive driving manoeuvres [9]. In a recent study, Wang et al. proposed an improved DTW-based approach for driving style recognition using smartphone sensor data [77]. These advancements in DTW-based methods enhance the accuracy and efficiency of driver behaviour classification.

Gaussian Mixture Models (GMMs): Gaussian Mixture Models have been applied to various aspects of driver behaviour analysis. Takeda et al. used GMMs for driver distraction detection [64], while Sathyanarayana et al. employed GMMs to estimate driving maneuvers [65]. GMMs have been applied in driver behaviour analysis to detect driver distraction, estimate driving maneuvers, and monitor road conditions. Almazán et al. utilized GMMs to monitor road conditions [43]. In a recent study, Liu et al. proposed a novel approach combining GMMs and Hidden Markov Models (HMMs) to recognize driving events based on vehicle sensor data [78]. These recent developments demonstrate the effectiveness of GMMs in modelling driver behaviours.

Support Vector Machines (SVMs): Support Vector Machines have been extensively studied for driver behaviour analysis. Van Leeuwen et al. and Sathyanarayana et al. used SVMs for driving event recognition and manoeuvre recognition, respectively [66, 67]. Li et al. applied SVMs for drowsiness detection [68], while Bhoraskar et al. used SVMs for vehicle state estimation [69]. In a recent study, Guo et al. proposed an SVM-based approach for driver drowsiness detection using multi-modal data fusion [79]. Another recent paper by Li et al. introduced a novel SVM-based framework for driver behaviour prediction using deep learning features [80]. These recent developments highlight the versatility and effectiveness of SVMs in capturing and analyzing driver behaviours.

Recent advancements in driver behaviour analysis have witnessed the emergence of hybrid methods that combine multiple techniques to leverage their strengths. One such approach is the integration of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks. Zhang et al. proposed a hybrid model that combines CNNs and LSTMs for driver drowsiness detection, utilizing the spatial and temporal features of driver-related data [73]. Another hybrid method involves the combination of Graph Convolutional Networks (GCNs) and traditional machine learning algorithms. Chen et al. introduced a driver behaviour analysis framework based on GCNs, enabling the exploitation of both spatial and relational information in driving scenarios [81]. These recent hybrid approaches highlight the potential of combining different techniques to enhance the accuracy and robustness of driver behaviour analysis systems. In these studies, machine learning techniques have been applied to various aspects of driver behaviour analysis, including classification of driver behaviours, modelling and prediction of driving manoeuvres, and detection of drowsiness. These recent developments showcase the potential of advanced machine learning methods in improving the accuracy and effectiveness of driver behaviour analysis systems.

2.2 Artificial Neural Networks

Artificial Neural Networks (ANNs) are mathematical models consisting of artificial 'neurons' that have input and output values and connections to other neurons, providing a framework for machines to learn and make decisions. The power of ANNs is derived from artificial neurons, each acting as a computational unit, receiving inputs and generating an output. These neurons are interconnected, forming a network capable of simulating complex learning tasks. A plethora of ANN architectures exist, each with its own distinct attributes, and this chapter will embark on a journey through a selection of these, particularly those pertinent to the work presented in this thesis.

2.2.1 Activation Functions

The pulse of a neural network lies in its activation functions. These mathematical entities transform the input data, enabling the network to learn and model complex, non-linear relationships. A variety of activation functions exist, each with its unique properties and use cases. Some of the popular ones: Sigmoid, Tanh, ReLu, Leaky ReLu, and Softmax, are showcased in Figure 2.1.

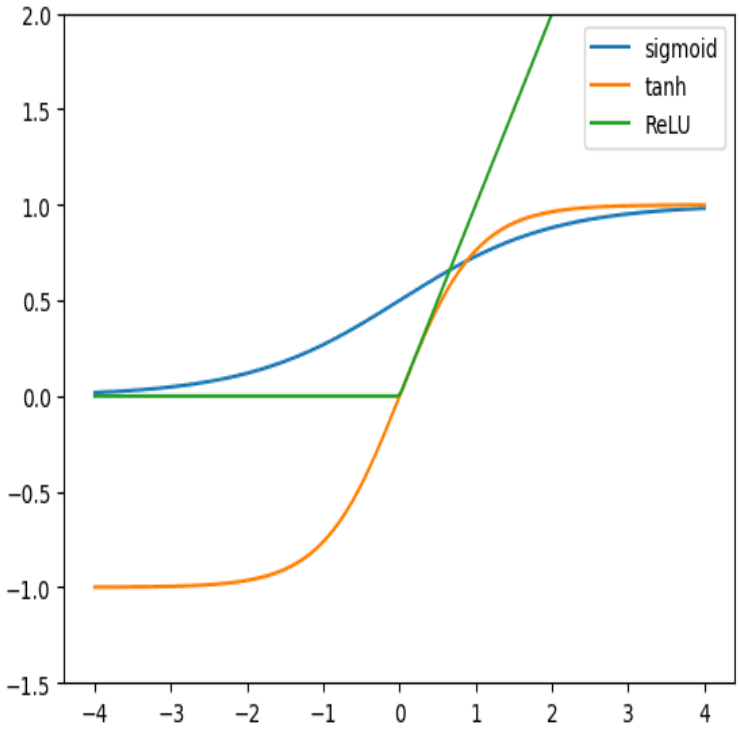


Figure 2.1: An Overview of Fundamental Activation Functions.

The Sigmoid Function: Logistic Transformation

The Sigmoid activation function, often termed as a logistic function, is defined by Equation (2.1). It squashes a real-valued input into a range between 0 and 1, making it an ideal choice for binary classification tasks. However, its use has been tempered due to the vanishing gradient problem, which hampers the learning of deep networks.

$$f_{sigma}(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

The Tanh Function: Zero-Centered Transformation

Tanh activation function, much like sigmoid, squashes the input but within the range of -1 and 1. Although it shares a similar shape to sigmoid, its zero-centred output makes it more preferred, as defined in Equation (2.2).

$$f_{tanh}(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.2}$$

The ReLu and Leaky ReLu: Linear yet Non-linear

Rectified Linear Unit (ReLu) has gained popularity for its simplicity and effectiveness, especially in deep networks. ReLu applies a simple threshold at zero, as defined in Equation (2.3). However, it's not without its flaws and thus was born its variant, the Leaky ReLu. This variant introduces a small slope for negative values to combat the dying ReLu problem, as shown in Equation (2.4).

$$f_{relu}(x) = max(0, x) \tag{2.3}$$

$$f_{lRelu}(x) = \begin{cases} x, & \text{if } x > 0 \\ ax, & \text{if } x <= 0 \end{cases} \tag{2.4}$$

The Softmax Function: Multi-Class Classification

The Softmax function is generally used in the output layer of a classifier where we are trying to attain the probabilities to define the class of each input. Softmax squashes a K-dimensional vector of arbitrary real values to a K-dimensional vector of real values in the range [0, 1] that add up to 1. It is defined in Equation (2.5).

$$f_{softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2.5}$$

2.2.2 Loss Functions

The last layer uses a "loss function," or "objective function," to estimate the quality of predictions of the network on the training data. A loss function calculates the difference between the prediction and the ground truth. There are several cost functions and two of the common ones are covered below. In addition, these functions are optimized in the learning phase of a CNN.

Mean Squared Error Function

The mean squared error (MSE) quantifies the difference between an actual and estimated value. The function is described as:

$$MSE = \frac{1}{2N}\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2 \tag{2.6}$$

where Y ,$\hat{Y}$ are the ground truth and predicted values, respectively.
The outlier in the data have a significant effect on this cost function because outliers create great error, and the squared error will be bigger.

Cross Entropy function

The cross entropy function H measures loss between true probability distribution $p(x)$ and outputted probability distribution $q(x)$ for variable x;

$$H(p, q) = -\sum_{x} p(x) \log q(x) \tag{2.7}$$

$$q(x) = \frac{exp(\hat{q}(x))}{\sum_{k} \hat{q}(k)} \tag{2.8}$$

Cross entropy has been generally preferred for classification while MSE is utilized for regression since cross entropy works well with probabilistic tasks [82].

2.2.3 Optimization

Searching for the minimum of a cost function is an optimization task while training a neural network. There are various types of algorithms for this optimization task:

Gradient Descent Algorithm

The procedure of repeatedly evaluating the gradient of a cost function and then performing a parameter update is called Gradient Descent. Gradient Descent (GD) is a basic algorithm that is developed to find the minimum of a cost function $H(\theta)$ around a set of parameters θ_t.

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta H(\theta_t) \tag{2.9}$$

By selecting a small enough η, the new θ makes the cost function $H(\theta)$ closer to the minimum. The most common optimizing Neural Network loss function is GD and also its derivations are used to build different kinds of algorithms. The first variation of GD is Stochastic Gradient Descent (SGD). It is created to deal with deep neural networks having large datasets.

Stochastic Gradient Descent Algorithm

GD recomputes gradients for similar cases before each update. Therefore, it applies inessential computations for large datasets. Stochastic Gradient Descent (SGD) performs one update at a time and it gets rid of this inessential computation. In SGD, a data sample (x^i, y^i) is chosen randomly at a time, and gradients are calculated on it. By this way, the optimization algorithm becomes feasible for large datasets. Updating frequently with high variance causes fluctuation on the objective function. SGD has converging issues due to being very stochastic. In order to deal with these issues, a larger subset of the data called as mini-batch can be used to stabilize the update.

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta H(\theta_t, x^i, y^i) \tag{2.10}$$

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta H(\theta_t, x^{i:i+n}, y^{i:i+n}) \tag{2.11}$$

Adaptive Moment Estimation (Adam)

Adam is another method that computes adaptive learning rates for each weight. The ADAM optimizer is an improved method for GD algorithm and some other techniques. The first difference is the usage of estimation instead of calculating the true gradient. The algorithm decreases computation time and cost. Using a running average of gradients and the square of the gradients helps ADAM to converge faster. The m_t and v_t zero biases are corrected with the following equations:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.12}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{2.13}$$

$$\hat{m}_t = \frac{m_t}{(1 - \beta_1^t)} \tag{2.14}$$

$$\hat{v}_t = \frac{v_t}{(1 - \beta_2^t)} \tag{2.15}$$

The update to θ_t is described by Equation (2.16). By dividing η by the moving average of the square of the gradient, a decaying effect is imposed on η and the update to t. ϵ is used as a small number in order to deal with division by zero issues.

$$\theta_{t+1} = \theta_t - \frac{\eta}{(\sqrt{\hat{v}_t} + \epsilon)} \hat{m}_t \tag{2.16}$$

Adagrad

The Adagrad algorithm learns the features with low occurrence by applying the learning rate η to each parameter θ_i separately. In Equation (2.17), g describes gradients and G_t is a diagonal matrix, where the learning rate is changed through. ϵ is selected as a very small number to prevent division by zero problems.

$$\theta_{t+1} = \theta_t - \frac{\eta}{(\sqrt{G_t} + \epsilon)} g_t \tag{2.17}$$

2.2.4 Back Propagation

Back propagation calculates the gradients needed for weight calculation to be utilized in the neural network. The backpropagation algorithm drives NNs to learn by minimizing the output error. The network error is calculated with a cost function. Equation (2.18) illustrates that the gradients $\frac{\partial H}{\partial w^l}$ and $\frac{\partial H}{\partial b^l}$, with given cost function H for a layer L can be calculated using the gradients from the layer $L+1$ via applying the chain rule. The error $\delta^{[l]}$ for the output layer l is described in Equation (2.18) .

$$\delta^{[l]} = \frac{\partial H}{\partial z^{[l]}} = \frac{\partial H}{\partial z^{[l+1]}} \frac{\partial z^{[l+1]}}{\partial a^{[l]}} \frac{\partial a^{[l]}}{\partial z^{[l]}} = \delta^{[l+1]} \frac{\partial z^{[l+1]}}{\partial a^{[l]}} \frac{\partial a^{[l]}}{\partial z^{[l]}} \tag{2.18}$$

$$\frac{\partial H}{\partial w^{[l]}} = \frac{\partial H}{\partial z^{[l]}} \frac{\partial z^{[l]}}{\partial w^{[l]}} = \delta^{[l]} a[l-1] \tag{2.19}$$

$$\frac{\partial H}{\partial b^{[l]}} = \frac{\partial H}{\partial z^{[l]}} \frac{\partial z^{[l]}}{\partial b^{[l]}} = \delta^{[l]} \tag{2.20}$$

For the last layer, the gradient is calculated differently as seen in Equation (2.21). It also varies depending on the choice of cost function and activation function in the last layer of the partial derivatives $\frac{\partial H}{\partial a^{[L]}}$ and $\frac{\partial a^{[L]}}{\partial z^{[L]}}$.

$$\delta^{[L]} = \frac{\partial H}{\partial z^{[L]}} = \frac{\partial H}{\partial a^{[L]}} \frac{\partial a^{[L]}}{\partial z^{[L]}} \tag{2.21}$$

2.2.5 Network Architectures

Various structures of Artificial Neural Networks (ANNs) have been designed, each tailored to process particular kinds of data or solve certain problems. Notable among these structures are the Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Autoencoders (AE), and Transformer Networks.

Multilayer Perceptron (MLP)

Multilayer Perceptrons (MLPs) represent the most fundamental form of artificial neural network. They are composed of three distinct types of layers: an input layer, hidden layers, and an output layer. Each layer is completely linked to the next layer, creating a comprehensive network. Known as a feedforward neural network, a MLP is perhaps the simplest neural network structure. It includes a minimum of three layers of nodes: an input

layer, one or more hidden layers, and an output layer. Every node in a layer maintains a connection with all nodes in the subsequent layer. MLPs are typically employed for simple classification and regression tasks.

Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are specifically designed to handle data structured in a grid pattern, like images that comprise a pixel grid. The term "convolutional" stems from the convolution operation which replaces the typical matrix multiplication in at least one of the network's layers.

Recurrent Neural Network (RNN)

Recurrent Neural Networks (RNNs) are created to work with sequence prediction tasks. They introduce state variables to hold previous data, together with current inputs, to determine current outputs. They are designed to operate with data sequences by integrating a type of memory that enables prior outputs to affect current inputs. This makes capable them for some particular tasks such as natural language processing, time series forecasting, and speech recognition.

Autoencoders (AE)

Autoencoders (AEs) are a type of artificial neural networks intended for unsupervised learning tasks, chiefly for reducing dimensionality and learning features. They are specifically designed to compress the input into a lower-dimensional code, and then reconstruct the output from this code. Autoencoders are a kind of neural network employed to learn efficient data representations, generally for the aim of reducing dimensionality or removing noise. They operate by converting the input into a compact representation and subsequently decoding this representation back to the original form. AEs can be utilized for tasks such as anomaly detection, image denoising, or collaborative filtering in recommendation systems. Autoencoders are an important part of unsupervised learning models in the development of deep learning. An Auto-encoder (AE) is a neural network architecture which copies its inputs to its outputs via reduced intermediary representation. The basic AE consists of two layers, encoding and decoding layers.

An AE can also be designed with multiple encoder and decoder layers. The AE tries to make outputs from a decoding layer the same as the inputs. This means that AE can

represent features of the input data in different dimensions by preserving essential information for reconstructing input data. Therefore, it can be beneficial for feature extraction. There are two types of autoencoder based on output dimension. If AE dimension is higher than the input dimension, it is named as over-complete AE and if AE dimension is lower than the input dimension, the name of AE is under-complete. An under-complete AE is often used as a feature reduction technique. The AE is usually used together with other data analyzing techniques that utilize data from the encoder layer of AE.

Sparse AE

A sparse autoencoder is a specific type of autoencoder, which is a neural network utilized for unsupervised learning of efficient data representations. Standard autoencoders are designed to learn data representations, often for the purpose of dimensionality reduction, by training the network to disregard insignificant data. The distinguishing characteristic of sparse autoencoders is the incorporation of a sparsity penalty into their training criterion, applied to the code layer of the autoencoder. This is implemented to prevent the autoencoder from simply mirroring its input. This sparsity constraint encourages the autoencoder to ensure that only a small number of neurons in the hidden layer are activated at any one time. Consider an autoencoder with input data y, hidden layer representation a, and output y'. The encoding weights and biases are W_1 and b_1, and for decoding are W_2 and b_2 shown in Figure 2.2.The forward pass of the autoencoder is calculated as follows [83]:

$$a = f(W_1 y + b_1),$$
$$y' = f(W_2 a + b_2),$$

where f is an activation function.

The autoencoder is trained to minimize a loss function, which includes a measure of the difference between the original input and the reconstruction (output). For sparse autoencoders, this loss function combines the standard reconstruction loss and a sparsity penalty.

The reconstruction loss, $L_1(y, y')$, measures the disparity between the input y and output y'. A common choice is the Mean Squared Error (MSE):

$$L_1(y, y') = ||y - y'||^2.$$

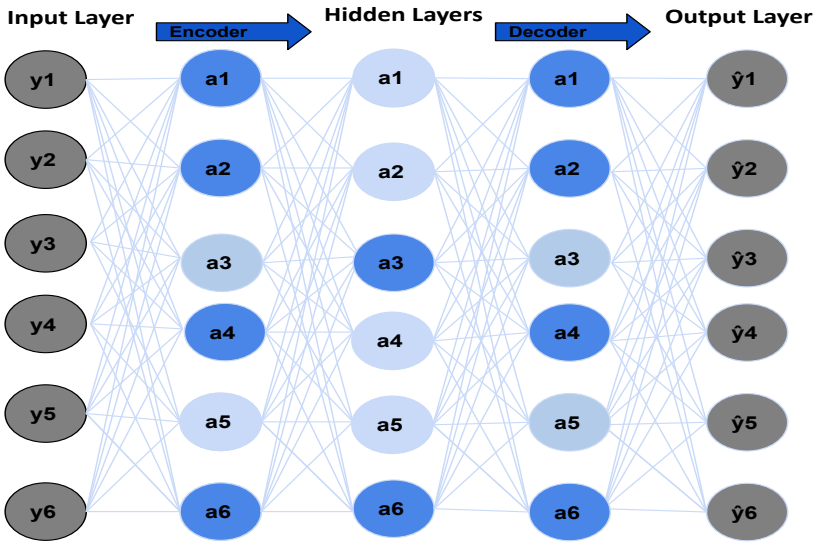


Figure 2.2: An example of Sparse AE

The sparsity penalty, $\Omega(a)$, quantifies the divergence between the desired activation level ρ and the actual average activation $\hat{\rho}$ of the hidden units. The Kullback-Leibler (KL) divergence is often used for this:

$$\Omega(a) = \sum(\rho \log(\frac{\rho}{\hat{\rho}}) + (1-\rho)\log(\frac{1-\rho}{1-\hat{\rho}})).$$

The total loss function, $L(y, y', h)$, is a combination of the reconstruction loss and the sparsity penalty, weighted by a factor λ:

$$L(y, y', h) = L_1(y, y') + \lambda\Omega(a).$$

The sparse autoencoder is trained by minimizing this total loss function, typically using methods like stochastic gradient descent or other optimization techniques [83].

2.3 Convolutional Neural Networks(CNN)

The Convolutional Neural Network (CNN) [84] is one of the most popular types of NNs for high-dimensional data, especially for images. CNNs work in a similar way to the typical NN covered in Section 2.2. CNNs are composed of neurons with adjustable weights and biases. The main difference is that each unit in a CNN layer is a high dimensional filter to be convolved with the input of that layer. Each neuron takes inputs with the kernel, applies a dot product and optionally follows it with a non-linearity. They have also a loss function on the last layer. It typically consists of alternating convolution and pooling layers. This type of NN application is capable of capturing patterns from high dimensional input data (images, texts and time series). By applying filters, the spatial context of input data can be represented similarly but in a smaller size. The reason for selecting CNNs for this study is that CNN has two main characteristics to learn features from time series data. These characteristics are **locality** and **weight sharing**. These features help to decrease the total number of learnable parameters. In this way, training and learning in each layer can be performed more efficiently.

2.3.1 Convolution Layer

A convolution layer is the main part of a CNN. It consists of filters (kernels) in order to apply convolution on given input data to produce output features.

Filters

Each filter is a grid having discrete numbers (weights). These numbers are learned while CNN is training. There are different approaches for weight initialization of filters such as; gaussian random, uniform random, orthogonal random and supervised pre-trained initialization. After the initialization of weight, weights are learned in a number of iterations by tuning based on given input and output.

Convolution Operation

Convolution is a mathematical operation that expresses a function for how the input I is modified to output S by applying kernel K to every pixel in I. For example, I represents a 2D input image with different colour channels (Red, Green and Blue-RGB), K is a kernel and s is the feature map after applying convolution operation in Equation 2.22. Since

convolution is commutative, Equation 2.22 can be rewritten as in Equation 2.23. The implementation in deep learning will be easier due to the lower variation of m and n. The Equation 2.24 is known as the cross-correlation function.

$$s(t) = (I \times K)(i,j) = \Sigma_m \Sigma_n I(m,n).K(i-m, j-n) \tag{2.22}$$

$$s(t) = (K \times I)(i,j) = \Sigma_m \Sigma_n I(i-m, j-n).K(m,n) \tag{2.23}$$

$$s(t) = (K \times I)(i,j) = \Sigma_m \Sigma_n I(i+m, j+n).K(m,n) \tag{2.24}$$

Convolution is used to detect features and create multiple feature maps for the identification or classification of the input vector. Applying multiple feature detectors identifies different kinds of features such as shapes, colors and bends for images.

A CNN's main difference is utilizing a convolution layer rather than using a fully connected layer. A fully connected layer can be defined as $y = f(x.w)$, where x is the input vector, y is the output vector, w is a set of weights, and f is the activation function. Accordingly, $y = f(s(x.w))$ can be a convolution layer description where s is the convolution operation between the input and the weights.

There are three key functional differences between the convolution layer and from fully connected layer. First of all, the convolution layer has less connection and weights than the fully connected layer when the same number of inputs and outputs are utilized. Secondly, the fully connected layer has much more interactions with inputs when compared with the convolution layer. The third difference is that parameter sharing in the convolution layer uses a smaller kernel and sliding on input. This means that fewer unique weights in convolution layers and less complexity exits.

2.3.2 Pooling Operation

The main idea in the pooling operation is that the exact location of the feature is not so important as long as the feature is discovered. Pooling function f is applied over parts of the input to create the output. The most common function f is the max operation (application to max pooling) and the output has a smaller dimension compared to the input. In other words, the operation can also be named as abstracting operation.

2.3.3 Statistical Analysis

In the context of comparing two classification methods, the paired t-test is a statistical tool used to assess the presence of a significant difference between our method and other meth-

ods. To conduct a paired t-test, differences between paired observations are calculated, and the mean and standard deviation of these differences are computed. A t-test is then performed to determine statistical significance. The results are interpreted by comparing the obtained p-value to a chosen significance level. A small p-value suggests a significant difference between the classification methods, while a large p-value indicates no significant difference. It's important to note that the assumptions of the paired t-test, particularly the normality of differences, should be carefully examined. The paired t-test is a valuable statistical tool for assessing differences between classification methods when applied correctly and with attention to its assumptions.

Key Assumptions

To ensure the reliability of a paired t-test, several fundamental assumptions must be met:

- The differences between the paired observations, representing the disparities in classification scores for each sample, should adhere to a normal distribution.
- The paired data should be structured so that each pair corresponds to similar or interrelated samples.

Formulation of Hypotheses

The paired t-test involves the formulation of two hypotheses:

- Null Hypothesis (H_0): There is no statistically significant difference between the two classification methods.
- Alternative Hypothesis (H_a): A statistically significant difference exists between the two classification methods.

2.4 Conclusion

This chapter provided related works, background algorithms and methods we will utilize for the driver behaviours analysis. There are still a number of challenges for future progress on driver behavior analysis in order to accomplish wide-scale deployment and use. These systems require many users and data before they become efficient. For this reason, there

should be a global dataset to test methods, and there should be global evaluation methods to select the best system. In the future, vehicle monitoring and driver assistance systems especially based on smartphones will be a crucial part of ITS because of their cost and adaptivity. In order to investigate further, more promising methods in the literature are ANN, Fuzzy Logic, HMM and SVM-based techniques. Three different ANNs methods were proposed in this study to analyze aggressive driver behaviour deeply. These are covered in the following Chapters 4,5,6.

Chapter 3

Stacked Sparse Auto-encoder Method to Select Driver Behaviour Features

3.1 Introduction

This chapter presents a deep learning method which is already been published as a conference paper to identify a low-dimensional feature set for driver action recognition [58]. The main purpose of this work is to obtain high classification scores of driver behaviours with lower computational complexity and lower storage requirements. To achieve these objectives, we focus on the representation of driver behaviour features. We propose a deep learning method similar to [85] to identify a low-dimensional feature set for driver action recognition. Unlike [85], we propose a different windowing method to generate the input data to the autoencoders, investigate the influence of the network depth, and evaluate our approach using smartphone data collected during actual on-road driving conditions.

By using multiple Sparse Autoencoders (SAE) and connecting them hierarchically, dimensionality reduction is achieved incrementally with each successive layer. In other words, using the hidden layers of information for each autoencoder, a reduced dimensionality representation is achieved, providing a more efficient feature abstraction. With this purpose, we implemented Stacked Sparse AutoEncoders (SSAE). Before the application of SSAE, we first normalize the data and reorganize it via a windowing technique. Finally, three-dimensional selected features are tested with several classifiers. An overview of the proposed method is shown in Figure 3.1.

The proposed method is verified by using SVMs, K Nearest Neighbour (KNN) and Naive-Bayes classification methods, with both the proposed low dimensional representation and with a Principal Component Analysis (PCA)-derived representation for comparison.

This chapter is organized as follows: In Section 2 we describe the proposed Stacked sparse autoencoder approach. Section 3 reports the experimental set-up and used dataset. In Section 4, selected method results on SVM, Naive-Bayes and K-NN are given and also compared with PCA method results. Section V concludes the chapter.

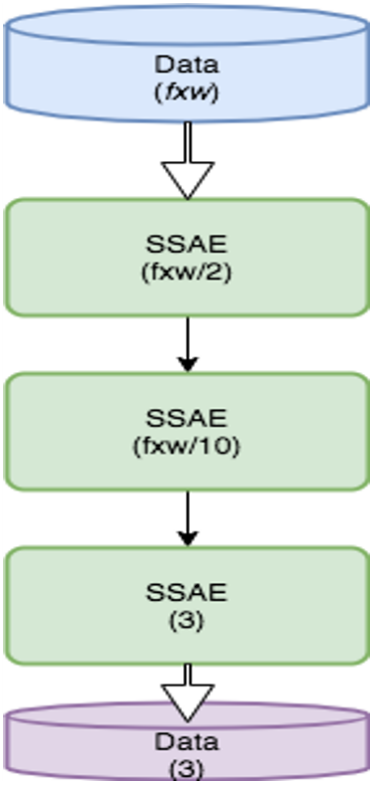


Figure 3.1: The Flow Diagram of the Method

3.1.1 Windowing

To generate the data windows, which are used as inputs into the AEs, a sequence for each feature is extracted, of the size equal to the defined window size. The features are then interleaved so that features which are sampled at the same time instant are adjacent to

each other, as illustrated in Figure 3.2. Following the windowing, the data is of dimension $windowsize \times featuredimension$. This dataset is then input into the SAE for dimension reduction.

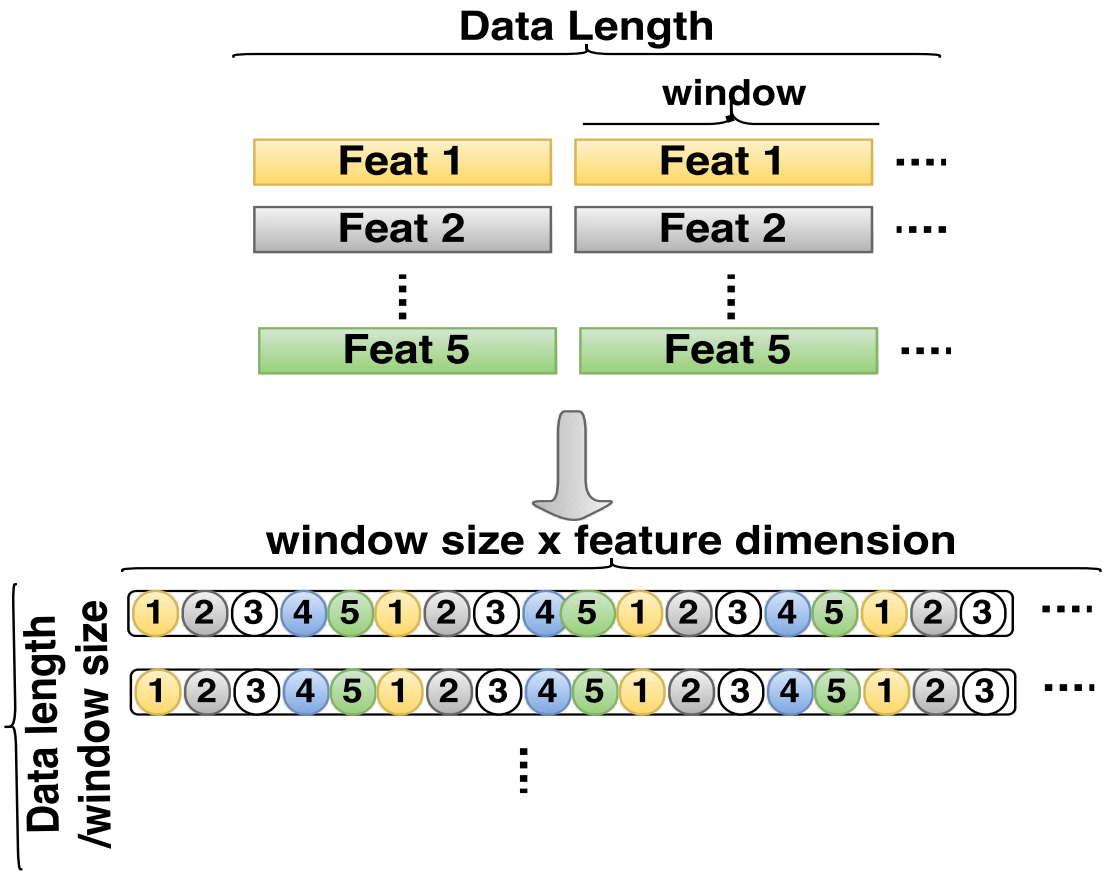


Figure 3.2: The Windowing Technique Application for Pre-processing

3.1.2 Sparse Autoencoders

Each AE consists of three hierarchical layers: the visible, hidden, and reconstruction layers. The SSAE consists of many SAEs, arranged hierarchically. An SAE is an AE where the training algorithm prunes connections between nodes to generate a sparse network.

Figure 3.1 shows the SSAE model and pre-processing used in this paper.

For the nth SAE used in the SSAE, we utilize the encoder function shown in Eq.(3.1) to generate the features of the hidden layer from the visible layers' features. Given an input, X, the network outputs a hypothesis $h(X_t^n)$, where W_e^n and b_e^n are the weight and bias parameters learned from the input [55]. Therefore, the input data acts as a training set to train the network to learn these parameters. The hypothesized output is determined with the following equation:

$$[h(X_t^n) = tanh(W_e^n X_t^n + b_e^n)] \tag{3.1}$$

Then, Equation (3.2) is used to reconstruct data:

$$[r(X_t^n) = tanh(W_d^n h(X_t^n) + b_d^n)] \tag{3.2}$$

where W_d^n represents the decoding weights and b_d^n is the bias for the decoding layer.

The Error Equation is:

$$\begin{aligned} E(X^n) = 1/(2N_X) \sum_{t=1}^{N_X} ||r(X_t^n) - X_t^n||_2^2 + ... \\ \alpha/2(||W_e^n||_2^2 + ||W_d^n||_2^2) + ... \\ \beta \sum_{i=1}^{D_h^n} KL(\Theta||\hat{h_i^n})||) \end{aligned} \tag{3.3}$$

A more complex neural network is designed by joining multiple neurons together so that the output data can be the input to any neuron. The neurons are organized into layers. After that, the decoder reconstructs the high-dimensional data. This procedure is repeated for all layers. Then, the final layer gives the activation of the deepest layer of hidden units. Sparsity is used to limit the activation of the hidden units with the goal of discovering the underlying structure of the data, regardless of the number of hidden units used. Θ represents the average activation of a particular hidden unit, resulting from all the inputs to that unit. In order to limit the weights for the encoder and decoder, the L2 norm is used as a penalty term in the error function. The term α affects the strength of the penalty term. To ensure sparsity in the hidden layer, an additional penalty term based on Kullback-Leibler divergence is added to the cost function. This divergence measures the difference between two probability distributions. β is used to control the strength of

the sparse term in the Kullback-Leibler (KL) equation Equation (3.4).

$$[KL(\Theta||h_i^n) = \Theta log(\frac{\Theta h_i^n}{\hat{h_i^n}}) + (1-\Theta) log(\frac{(1-\Theta)}{(1-\hat{h_i^n})})] \tag{3.4}$$

where Θ is a parameter specifying the sparsity of the target and $\hat{h_i^n}$ is the average value of the ith dimension. Unlike the standard SAE implementations that use the logarithmic hyperbolic function as the encode and decode function, we have selected the hyperbolic tangent function. These functions have similar saturation behaviour, but the tanh function has a range of $(-1, 1)$ and we want to keep the feature range wide. The KL equation needs hidden unit vector values to be between $(0, 1)$ so we mapped these values to this range from $(-1, 1)$. The remaining steps are the same as normal SAEs [86]. Lastly, in order to minimize the error equation, the backpropagation method is used to train the SAE. Then these SAEs are connected together by using the hidden layers of weight and bias information from each autoencoder's input.

3.2 Experimental Set-up and Dataset

To validate the proposed approach, seven real driving trips were recorded with a custom-developed smartphone app [87]. The trips covered a variety of routes in urban and suburban driving conditions, with the same car and driver. The smartphone's (Samsung Galaxy S4) GPS, gyrometer and accelerometer sensors were collected during each trip, with the smartphone placed in the vehicle cup holder. After data collection, all signals were interpolated to 100 Hz, a 0.2 rad/s threshold was applied to the gyro signal and all data normalized within their group to fit between (-1,1). Different window sizes were used to separate the data into windows. Figure 3.2 illustrates how the windowing technique:

- All times within the window are flattened column-base
- Features $\times$ window timestep becomes the dataset
- All models trained on that for different window start times

After windowing, the trips are labelled manually according to both GPS and gyro information, as illustrated in Figure 3.3. Ten possible labels have been defined, shown in Table 3.2:

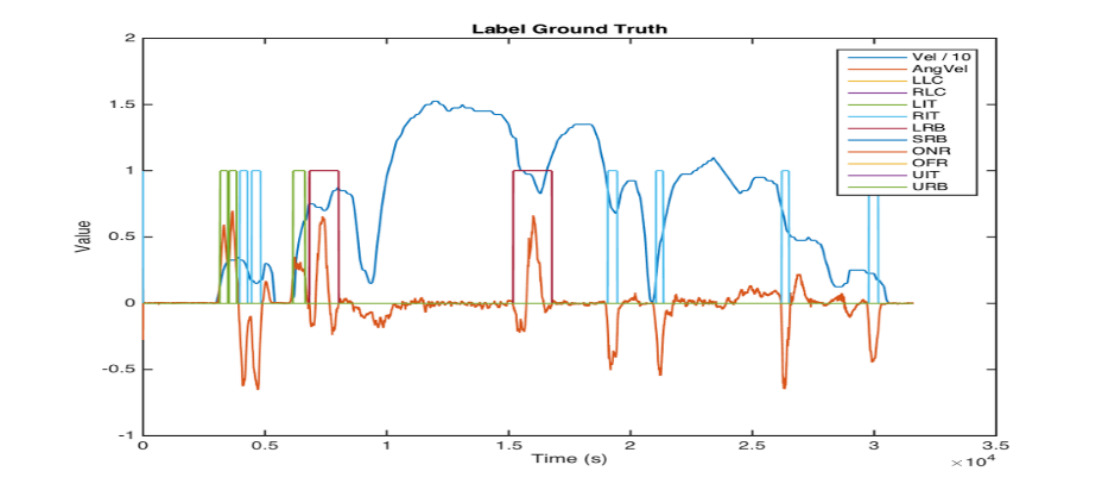


Figure 3.3: This Illustrates the Labelled Dataset Information

The data is then input to the SSAE for feature extraction. The following sensory information is used as the inputs:

- Linear velocity of the car $(x_1; t)$
- Acceleration magnitude of the car $(x_2; t)$
- Three dimensions of the gyroscope $(x_{3,4,5}; t)$

The output is set to generate three high-level features(y_t). This feature set size was selected to enable illustration.

3.2.1 SSAE Implementation

The AE parameters were set to $\alpha = 0.15, \beta = 0.7$ and $\Theta = 0.9$ and high-level features were estimated for different window sizes, using $10, 20, 50$ and 100 samples per window. To generate the high-level features, we applied the 3-layer SAE with hidden layer sizes of

$w \times D/2$, $w \times D/10$ and 3, where w is the window size and D is the dimension of the data after windowing. The extracted three-dimensional features were visualized by line plotting. Figures 3.4 and 3.5 and 3.6 illustrate the results for window sizes $10, 50$, and 100, respectively.

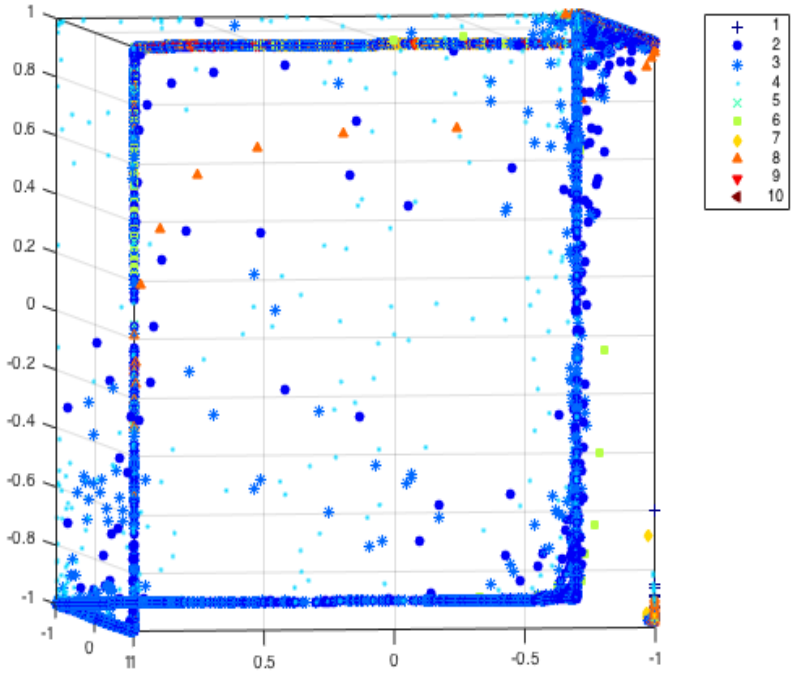


Figure 3.4: The 3D Illustration of Features for Window Size 10

Colours and marker types represent Class 1 to Class 10 respectively (as described in Table 3.2)

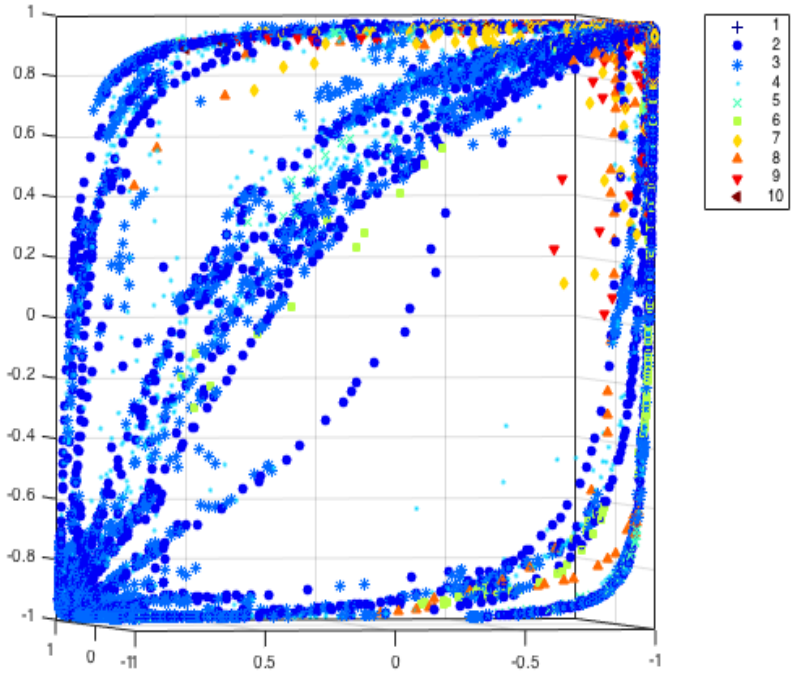


Figure 3.5: The 3D Illustration of Features for Window Size 50

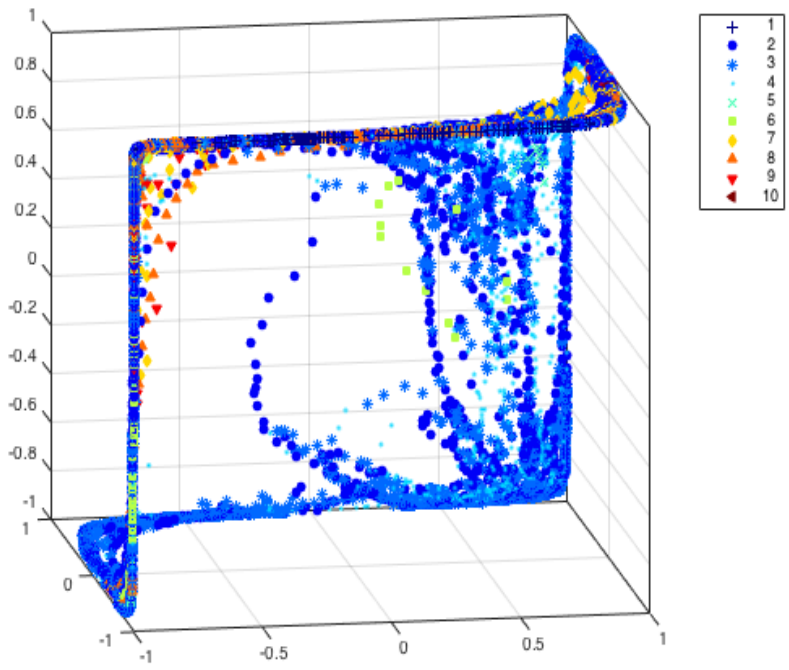


Figure 3.6: The 3D Illustration of Features for Window Size 100

From visual inspection, the 50 and 100 window sizes feature appears to produce better separation between features. In other words, it is easier to differentiate classes by using these three features.

We next compare the proposed approach with Principal Component Analysis (PCA) [88], a dimensionality reduction method. PCA is applied to the 100 window size data and the feature dimension is reduced from 5×100 to 3. Figure 3.7 shows these three-dimensional features.

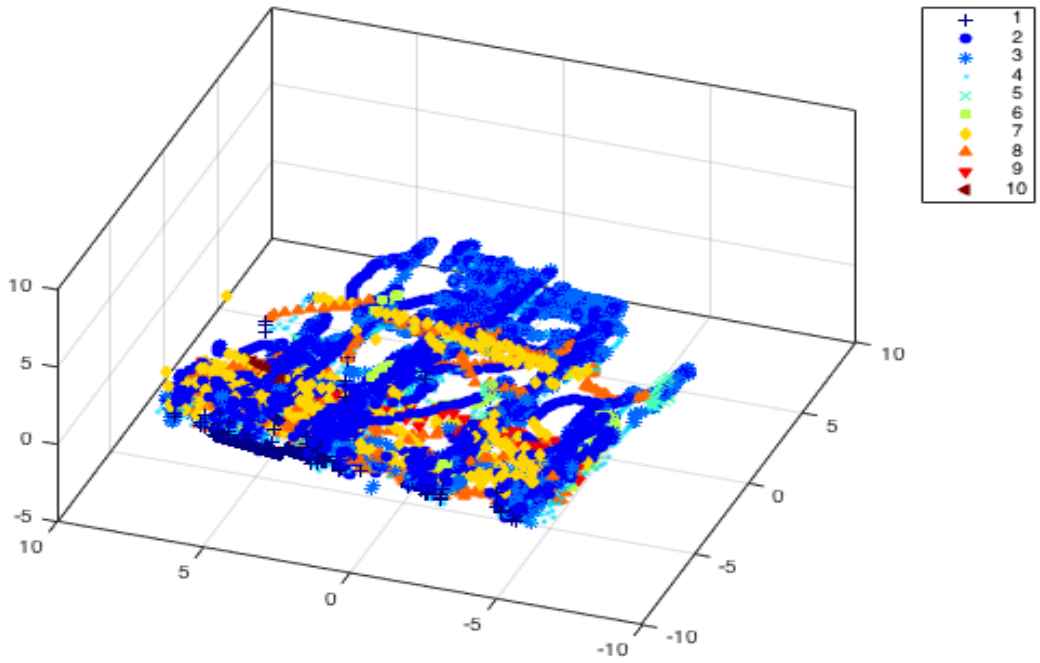


Figure 3.7: The 3D Illustration of Features which are Selected by PCA
Colours and Marker Types Represent Class 1 to Class 10 Respectively (as described in Table 3.2)

3.3 Classification Results

In order to evaluate the relevance of the high-level features, classification methods can be applied on these features. Using the manual data labels, we apply SVM, K-NN and Naive-Bayes supervised classification techniques and compare classification performance using the SSAE and the PCA-generated low-dimensional features. For training the SVM, Leave-one-out sampling and Radial Basis functions are used, as implemented in the Libsvm toolbox [89].

Tables 3.3, 3.4 and 3.5 show the confusion matrices for varying window size, from 10, 50 to 100, respectively. In the tables, each row corresponds to the instances in an actual class while each column corresponds to instances in a predicted class. The action labels are shown in Table 3.2. In general, increasing the window size from 10 to 100 improves

Table 3.1: Accuracy and F1 Score

#of SAE	Window size	Classifier	Accuracy [%]	F1 score
3	100	SVM	75	0.87
3	50	SVM	71	0.86
3	20	SVM	70	0.84
2	50	SVM	69	0.82
3	100	NB	68	0.72
2	20	SVM	67	0.74
4	20	SVM	67	0.70
1	50	SVM	65	0.72
1	20	SVM	65	0.70
3	100	K-NN	65	0.68
6	50	SVM	62	0.64
6	20	SVM	60	0.63

Table 3.2: Labels

CLASSES	ACRONYMS	MARKERS TYPES
1-Stop	STOP	PLUS SIGN
2-Acceleration	ACC.	ROUND
3-Coasting	COAST.	ASTERISK
4-Deceleration	DEC.	POINT
5-Left Lane Change	LLC	CROSS
6-Right Lane Change	RLC	SQUARE
7-Left Intersection Turn	LIT	DIAMOND
8-Right Intersection Turn	RIT	UPWARD-POINTING TRIANGLE
9-Entering Roundabout	ENR	DOWNWARD-POINTING TRIANGLE
10-Exiting Roundabout	EXR	LEFT-POINTING TRIANGLE

Table 3.3: Confusion Matrix for Window Size 10 (in %)

	STOP	**ACC**	**COAST**	**DEC**	**LLCH**	**RLCH**	**LIT**	**RIT**	**ENR**	**EXR**
STOP	98.00	0.20	0.00	0.28	0.00	0.00	0.00	0.00	0.00	0.00
ACC	1.80	50.70	24.68	04.05	0.00	0.00	1.65	1.63	0.00	0.00
COAST	0.11	12.32	85.91	12.58	0.82	0.00	2.33	0.51	1.33	0.00
DEC	5.30	3.66	21.92	54.12	0.55	0.00	1.16	2.35	2.33	0.00
LLCH	0.02	0.22	0.63	1.18	16.99	0.00	0.00	0.00	0.67	0.00
RLCH	0.00	1.45	0.55	1.99	0.82	4.77	0.00	0.82	0.00	0.00
LIT	0.00	0.58	0.08	0.27	1.37	0.00	87.09	0.10	7.33	0.00
RIT	0.16	0.82	0.00	0.46	0.00	0.00	0.00	83.46	4.33	0.00
ENR	0.00	0.56	0.06	0.26	0.00	0.00	3.40	2.04	46.67	0.00
EXR	0.00	0.14	0.00	0.10	0.00	0.00	1.07	1.43	0.00	5.45

Table 3.4: Confusion Matrix for Window Size 50 (in %)

	STOP	**ACC**	**COAST**	**DEC**	**LLCH**	**RLCH**	**LIT**	**RIT**	**ENR**	**EXR**
STOP	96.78	0.22	0.09	0.87	0.00	0.00	0.00	0.00	0.00	0.00
ACC	4.14	52.89	17.93	12.83	1.40	0.92	2.21	1.00	0.83	4.54
COAST	2.13	9.02	89.40	9.00	4.89	0.00	0.98	1.24	0.00	0.00
DEC	11.39	9.73	16.55	55.21	2.10	1.39	0.98	2.488	0.00	9.10
LLCH	0.00	0.31	0.37	0.67	48.25	0.00	0.00	0.00	0.00	0.00
RLCH	0.00	1.35	0.43	0.71	0.00	40.74	0.00	0.00	0.00	0.00
LIT	0.52	1.66	0.76	1.80	0.00	0.00	42.89	1.24	0.00	4.54
RIT	2.13	1.33	0.41	1.78	1.40	0.00	0.98	46.02	0.00	0.00
ENR	0.17	0.62	0.09	0.96	0.00	0.00	0.49	0.25	30.00	0.00
EXR	0.00	0.22	0.01	0.02	0.70	0.00	0.00	0.00	0.00	40.91

Table 3.5: Confusion Matrix for Window Size 100 (in %)

	STOP	**ACC**	**COAST**	**DEC**	**LLCH**	**RLCH**	**LIT**	**RIT**	**ENR**	**EXR**
STOP	97.70	0.48	0.00	0.40	0.00	0.00	0.00	0.00	0.00	0.00
ACC	5.66	58.69	14.94	12.18	1.37	0.00	0.48	1.53	0.00	0.00
COAST	2.42	6.12	90.95	9.16	0.00	0.00	0.97	0.51	0.00	0.00
DEC	15.93	7.89	13.42	61.40	0.00	0.92	0.00	1.50	0.00	0.00
LLCH	0.00	0.22	0.29	0.44	63.01	0.00	0.00	0.00	0.00	0.00
RLCH	0.11	0.79	0.32	0.58	0.00	58.72	0.00	0.00	0.00	0.00
LIT	0.35	1.72	0.37	1.11	0.00	0.00	59.71	0.51	0.00	0.00
RIT	0.69	0.93	0.17	1.42	0.00	0.00	0.90	65.82	0.00	0.00
ENR	0.00	0.22	0.02	0.93	0.00	0.00	0.97	0.00	51.67	0.00
EXR	0.00	0.04	0.02	0.04	0.00	0.00	0.48	0.00	0.00	63.64

Table 3.6: Confusion Matrix for PCA (in %)

	STOP	**ACC**	**COAST**	**DEC**	**LLCH**	**RLCH**	**LIT**	**RIT**	**ENR**	**EXR**
STOP	99.65	0.09	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00
ACC	1.28	52.16	22.80	5.82	0.00	1.83	3.40	0.51	6.67	0.00
COAST	0.46	13.48	86.28	10.27	2.74	5.50	1.46	2.56	5.00	0.00
DEC	6.24	6.00	20.39	53.85	6.85	3.67	0.98	3.06	0.00	0.00
LLCH	0.00	0.22	0.42	0.76	46.57	0.00	0.00	0.00	0.00	0.00
RLCH	0.00	1.12	0.51	0.58	0.00	45.87	0.00	0.00	0.00	0.00
LIT	0.00	1.90	0.39	0.76	1.37	0.00	60.19	3.06	3.333	0.00
RIT	0.35	1.32	0.15	2.04	0.00	0.00	3.89	50.51	6.67	0.00
ENR	0.00	0.79	0.10	0.58	0.00	0.00	1.46	1.02	33.33	0.00
EXR	0.00	0.22	0.00	0.04	0.00	0.00	0.00	0.51	0.00	36.36

classification performance, but the influence of window size differs between classes. Smaller window size gives better performance for the stop, left and right intersection turn classes. However, if the window size is too large, multiple actions can occur in the same window, and the classification performance rapidly decreases, thus the best window size is 50 or 100. Comparing the SSAE features to the PCA-generated features, PCA has the highest score for class 1, which is stop, but SSAE performs better for most other classes given the same window size. K-NN and Naive Bayes classifiers were also applied to the same dataset and gave very similar and slightly less accurate results than the SVM results above. The overall scores are illustrated in Table 3.1.

The number of SAE layers and the data sources used in the deep-learning structure are the main differences between our proposed method and Lui et. al. [85]. The number of layers of SAE has a significant impact on both the reduced features and the classification results. While using just one SAE gives 65% overall accuracy, utilizing three SAEs results in 70% overall accuracy with SVM and 20 sample window sizes. On the other hand, with six SAEs for the same window size and classifier, 60% accuracy is obtained. These results are shown in Table 3.1. Therefore, selecting the correct number of SAE layers significantly impacts performance. An increase in the number of SAEs not only increases training time and computational cost but can also cause over-fitting. Furthermore, the window size also has an effect on feature reduction and classification accuracy. For example, while window-size 10 is effective in detecting intersection turns (both left and right), window sizes 100 and 50 are better for detecting lane changes (both left and right) and roundabouts (both left and straight), as illustrated in Tables 3.3,3.5,3.4.

3.4 Conclusion

In this chapter, we proposed a feature reduction system for identifying features that differentiate human driving behaviours based on sparse autoencoders. The proposed method was tested with smartphone-collected data. The experiment illustrates that the model is capable of capturing features that are helpful in discriminating between driving manoeuvres. This is an important first step for designing driving assist systems that can operate in real-time using embedded smartphone sensors and limited computational resources. As future work, automated numbers of SAEs and parameter determination algorithms can be created as suitable for each data type. This enables to use SSAEs in the most efficient way.

Chapter 4

Parallel CNN Classification for Aggressive Driver Behaviour

4.1 Introduction

This chapter presents a deep learning method which has already been published as a conference paper to classify aggressive and normal driver behaviours [59]. The study of driving behaviour analysis is an increasingly important field in the context of prioritizing road safety. This is due to the growing requirement for sophisticated systems that can assist drivers and automated vehicles in navigating safely. However, the cost of modifications on infrastructure generally is high and the results of these modifications must be illustrated in detail. Therefore, data simulators can provide a low-cost solution to model situations in a realistic manner by predicting the consequences of infrastructure modifications at first glance. A microscopic traffic simulator, called SUMO, and a robotic simulator, Webots, are chosen for this study since they provide the ability to model vehicles individually and to control their features, including driving behaviours. For the purpose of learning to identify aggressiveness in driving, we first started to train and test the proposed model in this study on a **simulated dataset** and then on a newly collected real-world driving resulting in the Driver Behaviour Learning (DBL) dataset. We intend to derive the effect of aggressive personality parameters on vehicle dynamics by using a deep learning method, namely CNNs working in parallel. Our main contributions are:

- Introduction of automated data production by simulators with different driver features.

- Introduction of real-world driving data with different driver features and labels.
- Deriving the impact of aggressive personal features by classification.
- Developing a parallel CNN architecture that utilizes a common early stage and a split-stage training for two related tasks to support each other task.
- Adapting the method to the real-world DBL driving data.

This chapter is structured as follows; the tools used in the development of this work are covered in Section 4.2, section 4.3 describes how the dataset is simulated and collected with different features including driver's behaviour and explains the developed architecture, assumptions and experiments, Section 4.4 presents the experimental results on the simulated and real-world driven datasets, and finally we present the final conclusions in Section 4.5.

4.2 Tools

In this study, we use the system architecture shown in Figure 4.1 to study the aggressiveness of drivers using a map tool (OSM), a traffic simulator (SUMO), a robotic simulator (Webots), real-driven Driver Behaviour Learning dataset (DBL dataset) and a deep classification tool (PCNN). The main components of the aggressiveness driver analysis system are a map, a traffic simulation, a controller and a deep classification tool (see Fig 4.1).

Our implemented data simulation tool has five main modules:

- Open Street Map (OSM): creates a random area from World and downloads OSM map information for this area.
- Agent Behaviour: includes all features and actions related to each individual agent.
- Network: contains the relevant information of the network (the layout, the distances and time step occupancies).
- SUMO: it is responsible for getting a map for the network, parsing the network file, creating and saving the network and its population in order to communicate with Webots, and then save the simulation results to a file

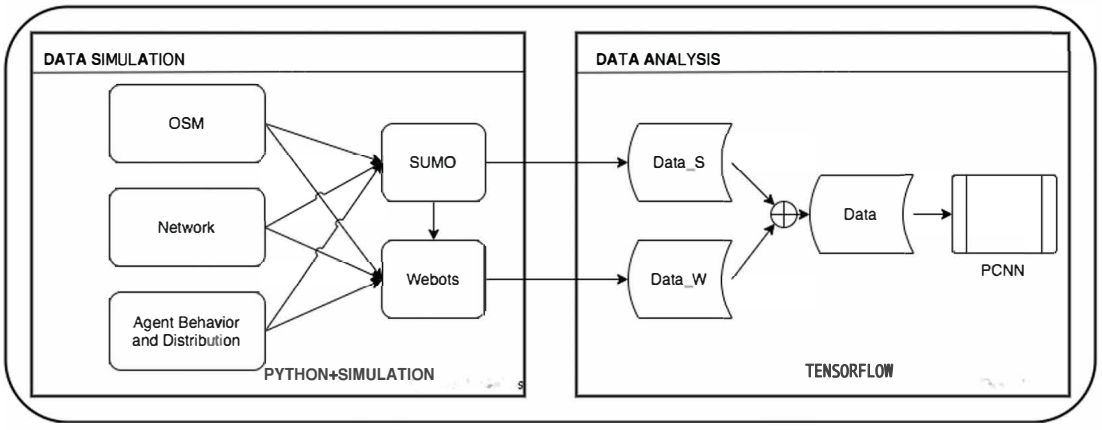


Figure 4.1: General System Architecture for Driver Behaviour Analysis

- Webots: creates the vehicles, parses the route files and attributes traffic and vehicle position information from SUMO generates sensors (GPS, accelerometer, gyro meter etc.) outputs.

4.2.1 SUMO Simulation Tool

In order to simulate driver behaviours, we need to not only simulate the traffic but also allow control of the vehicle features and behaviours. Simulation of Urban Mobility (SUMO) [90] meets this study's requirements. SUMO is an open-source multi-modal traffic simulation tool. Multi-modal means that different kinds of vehicles are included in this simulator. Furthermore, SUMO is a type of microscopic simulator, indicating that each vehicle moves its own route, and behaves individually. SUMO has a continuous space, and it records data for each vehicle in discrete time (the default time step is one second). It also includes a traffic lights implementation option. We have used SUMO version 0.12.3 in this study, its Gui-interface is illustrated in Figure 4.3.

The vehicle dynamics of driving on the road in SUMO are determined by several models. Three of the most important models for driver behaviours are:

- **Car-following Model:** controls a vehicle's speed according to the vehicle ahead of it.

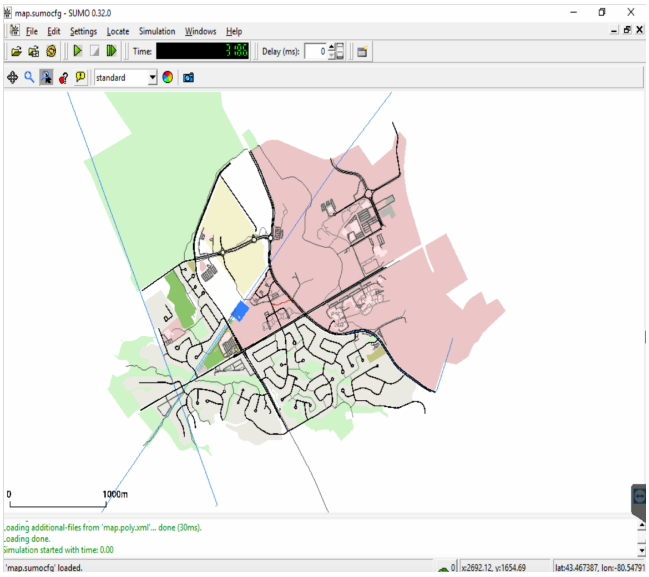


Figure 4.2: An Illustration of SUMO-gui

- **Intersection Model:** controls vehicles' decisions at different types of intersections based on right-of-way rules and gap acceptance.
- **Lane-changing Model:** determines lane position on multi-lane roads and adjusts speed for lane changing.

Lane-changing Model

The lane-changing model consists of two main functions. First, it decides the actions for each vehicle for a single step according to its own route and surrounding vehicle conditions. Then, it adjusts vehicle velocity according to the vehicle ahead to execute the desired lane change. The sub-actions at each time step for every vehicle are:

- Compute 'BestLanes' which are preferred as the following lane.
- Compute speed for staying on the current lane safely and integrate with speed for lane changing action from the previous simulation step.
- Compute change request, which can be resulted as left, right or stay.

- Based on calculation either do lane-changing manoeuvre or recompute speed request for the following simulation step.

There are four different lane-changing types in this model based on changing reasons. These are strategic lane change, cooperative lane change, tactical lane change and regulatory lane change.

Car-following Model

The car-following models are designed to control vehicles' longitudinal behaviour by describing the interactions of vehicles with surrounding vehicles (both the follower and leading vehicles). The SUMO car-following model was developed by Krauss in 1997. It is based on the calculation of a safe speed. The basic safe speed calculation is as follows:

$$V_{safe} = V_l(t) + \frac{g(t) - V_l(t)T}{\frac{V_l(t)+V_f(t)}{2b} + T} \tag{4.1}$$

where $V_l(t)$ denotes the leading vehicle speed in time t, $g(t)$ means the leading vehicle in time t, T is the driver's reaction time and b represents the the vehicle's maximum deceleration, $V_f(t)$ is the speed of following vehicle .

$$V_{des} = min[V_f(t) + a_{max}T, V_{max}, V_{safe}] \tag{4.2}$$

V_{safe} may be calculated as larger than the maximum allowed speed or larger than the vehicles' capacity. The minimum of these values is selected as the desired speed.

SUMO parameters

There are many parameters in SUMO to arrange more realistic trip simulations. In this part, we will focus on several important parameters that affect driver behaviours in particular. We associate the following SUMO behavioural parameters from the two above models, the car-following model (Krauss) and lane-changing model (SL2015):

- **accel:** vehicle's acceleration ability / The higher, the more aggressive the driver.
- **decel:** vehicle's deceleration ability / The higher, the more aggressive the driver.

- **minGap:** The minimum gap acceptance parameter allows us to simulate the tailgating behaviour / The less the gap between the lead vehicle and the current vehicle, the more tailgating effect produced.
- **maxSpeed:** the vehicle's maximum speed ability / The higher the speed, the more aggressive the driving is.
- **sigma:** a driver's imperfection / The lower the value the more aggressive the driver.
- **tau:** a driver's reaction time / It is lower in more aggressive drivers.

The lane-changing behavioural parameters are:

- **lcImpatience:** willing to impede vehicles with a higher priority / The higher, the more aggressive driver behaviour
- **lcAssertive:** willing to accept smaller longitudinal gaps in the required gap is divided by this value / The higher, the more aggressive behaviour

4.2.2 Webots

Webots is an efficient open-source robotics simulation software using the Open Dynamics Engine (ODE) library. Vehicle dynamics and physics can be customized in Webots based on ODE. In other words, Webots provides a real-car-based vehicle model with realistic dynamics features with ODE. Also, multiple intelligent vehicles can be simulated in Webots. In the Webots simulation world, the Open Street Map (OSM) data is used to extract the road network and its features. The OSM maps are converted to a graph data structure for Webots. Another extension of Webots is communication with SUMO to create traffic for simulation. SUMO simulates traffic and controls each vehicle's behaviours, while Webots simulates each vehicle's dynamics, enables the addition of sensors into vehicles and calculates outputs of these sensors based on coming traffic information from SUMO. We use Webots to build GPS, gyro-meters, accelerometers and GPS sensors on vehicles. The gathered information from these sensors is combined with SUMO output in order to label the dataset.



Figure 4.3: An illustration of Webots

4.2.3 Keras and Tensorflow

Keras [91] is a free deep-learning framework for Python. It provides an efficient way to define and train many deep-learning models. It was developed specifically to allow researchers to implement their experiments quickly. However now, the number of Keras users has increased rapidly. Keras users include researchers, engineers students etc. Many well-known companies such as Netflix, CERN, Yelp, Google and Uber use Keras for studying various problems [92].

The main key features of Keras are as follows:

- Provides the code to run efficiently on CPU or GPU.
- Consists of different kinds of deep methods such as; convolutional networks, recurrent networks, and combinations of both.
- Supports many types of network architectures such as multi-input or output models, layer sharing, model sharing etc.

Keras allows different backend engines to work together such as TensorFlow, Theano, and Microsoft Cognitive Toolkit (CNTK). Tensorflow uses data flow graphs for numerical computation. Treating all operators as nodes is the main difference of Tensorflow since this guides users to build neural networks in an intuitive way [92]. Another difference is the TensorBoard tool which visualizes neural networks via tracking of all parameters while training.

Behavioral Types	Manoeuvre Types
Normal/Safe	Right Turn
	Left Turn
Aggressive	Right Lane-Change
	Left Lane-Change

Table 4.1: Type of Classes

4.3 Experimental Setup and Dataset

In this work, we study an important real-world problem of driver behaviour analysis and propose a deep learning-based solution due to the complexity of the task. To validate the proposed approach, the data were simulated by combining a micro-traffic simulator, SUMO and a robotic simulator, Webots and the method is applied to this simulated data, a small real dataset collected via smartphone and big real-world driven Driver Behaviour Learning dataset. Initially, our proposed approach is validated on simulations, a micro-traffic simulator SUMO and a robotic simulator Webots. Then the method is also applied to the DBL dataset. During simulation runs, many trips were recorded with different parameters which control aggressive driver behaviours. Also, the trips cover a variety of real map routes in urban and suburban driving conditions with different types of cars and different kinds of drivers. A range of window sizes was used to extract a time series data points using a sliding window approach. Each driver type will be assigned a specific value for minimum gap acceptance, reaction time, acceleration and deceleration rates, maximum speed etc. Position, velocity and angle information were saved during each trip at 100 Hz. 4-sec, 3-sec and 2-sec window sizes were used as sliding windows to separate the time series data into windows. After windowing, the trips are labelled automatically according to simulator features in terms of behaviour (normal/safe, aggressive) and manoeuvre (right turn, left turn, right lane change, left lane change). Aggressive vs. normal driver behaviours are simulated by tuning the SUMO parameters 4.2 based on documentation and previous studies [93][94][95]. There are different types of label classes based on manoeuvre types and behaviour types. In total, 6 initial labels are defined for two tasks in Table 4.1. The number of classes increases to 8 if we combine two tasks.

4.3.1 Simulated Dataset

The dataset simulation steps in Figure 4.1 are described in the following steps. These steps can be grouped into two phases: **the generating phase and the application phase**. In the generating phase, we require the city's latitude and longitude list, internet connection, OSM, SUMO and Webots.

- Open Street Map is downloaded online by selecting a random latitude and longitude from the world city latitude and longitude list.
- Network is created from OSM in order to be utilized for SUMO.
- Total number of vehicles is defined randomly from a defined range.
- Different vehicle types were defined for different driving behaviours (aggressive and normal) by changing parameters based on previous works parameters[93][94].
- Then, the probability of these vehicle types is distributed randomly.
- Random trips are created by selecting random two nodes for start and end points and by utilizing numbers and the distribution of vehicles.

The second phase is the application of all information which are random trips, vehicle types and numbers into the simulator:

- SUMO utilizes network, trips and vehicle type information to simulate traffic and save and transfer data to Webots.
- Webots use network, trips and traffic information to simulate vehicle and sensor dynamics and save their output data.
- SUMO and Webots time series data are combined together.
- Finally, time window labels can be created successfully according to output data and used parameters of SUMO simulation.

Aggressive vs normal driver behaviours are simulated by tuning these parameters based on SUMO instructions and previous studies [93][94][95]. Each driver type will be assigned a specific value for minimum gap acceptance, reaction time, acceleration and deceleration rates, maximum speed etc., which are described in Table 4.2.

Parameters	Normal Driver Behaviour	Aggressive Driver Behaviour
acc(m/s^2)	2.6	3.2
decel(m/s^2)	4.5	5.5
sigma($0-1$)	0.5	0.3
maxSpeed(m/s)	23	35
minGap(m)	2.5	0.9
tau (s)	1.5	1
impatience($0-1$)	0	1
lcAssertive($0-1$)	0	0.9

Table 4.2: SUMO Parameters for Aggressive and Normal Driver Behaviours

Simulation data is generated according to randomly selected region road map information, random vehicle distributions and determined driver and vehicle parameters. The output of the simulation includes many features, but only position, velocity and angle information have been utilized to apply the proposed method. The dataset consists of variables which are time series and uses sliding windows applied on these time series. Some samples of data are shown in Figure 4.5 and 4.6. We use the data extracted from these time windows to predict these classes. This time series dataset used are the **plain, rotated and converted** formats which are mentioned in Data Pre-processing Section 4.3.1. PCNN layers' input dimensions, kernel and pooling window sizes change based on the input data format. These can be found in the Tables 4.3 and 4.4 for both plain or rotated and converted time series datasets.

Data Pre-processing

- **Plain time series without reformatting:** 2D Data output taken from directly simulators. Dimensions are time in the time window (300ms) and vehicle features (speed, x, y, angle) from simulations (4x300 for just one manoeuvre sample).
- **Time series with rotated location x and y:** Zero-centred x and y position data has high spatial variance within a single class. In order to translate this into a more informative format for any application method, we rotate each (x,y) datapoint so that they are all in the first and second quadrants of the x and y coordinate plane. Essentially we normalized the initial direction for each manoeuvre to North. Figure 4.5 illustrates positions before and after the application of rotation.

- **Image map data converted from time series:** For comparison purposes, images were created from time series data. The steps for creating images are shown in the Algorithm 1. Figure 4.4 shows this conversion step by step.

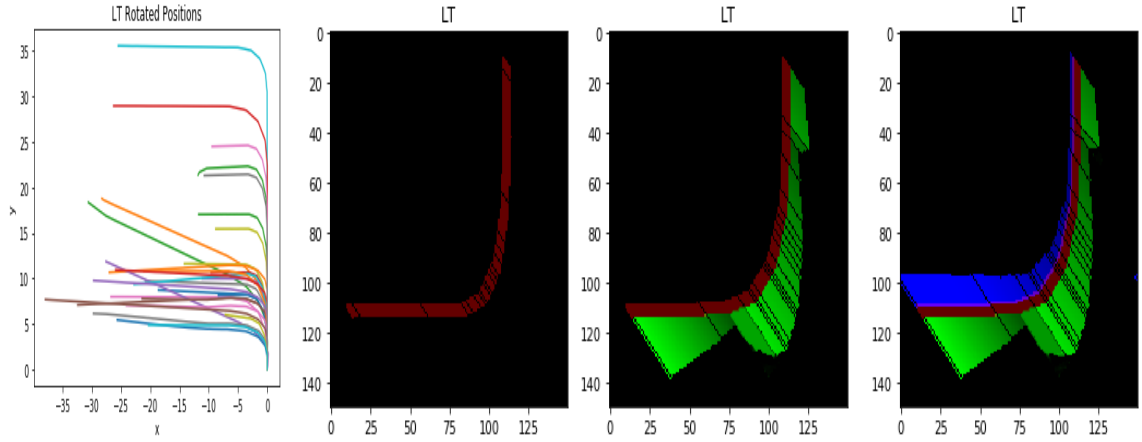


Figure 4.4: A Rotated Left Turn Time Sequence Sample Conversion Steps to Image

Red is x and y locations are represented as pixels locations with a given thickness. Then, velocity is added upon this pixel locations as a green colour (amplitude and brightness is changing according to velocity). Finally, angle data is subtracted from pixel locations, and is shown in blue colour (just amplitude changes based on angle data)

Algorithm 1 Image Map Conversion from Time Series Algorithm

1: **procedure** $(n, x, y, features)$
2: Create empty $[feature_{numbers} \times N \times N]$ image
3: Read the values of x and y position of vehicle data
4: Normalize x and y values and re-scale between 0 and (N-10) to place values into pixels of image(to fit into the image)
5: Use x and y data as pixel locations of new image to plot a map
6: Use another feature (i.e. speed) for colouring of these pixel locations by adding value of features to pixel positions
7: Use another one (i.e. angle) in order to give thickness to the plotting by utilizing feature values as thickness parameters by subtracting the value of features to pixel positions.
8: **end procedure**

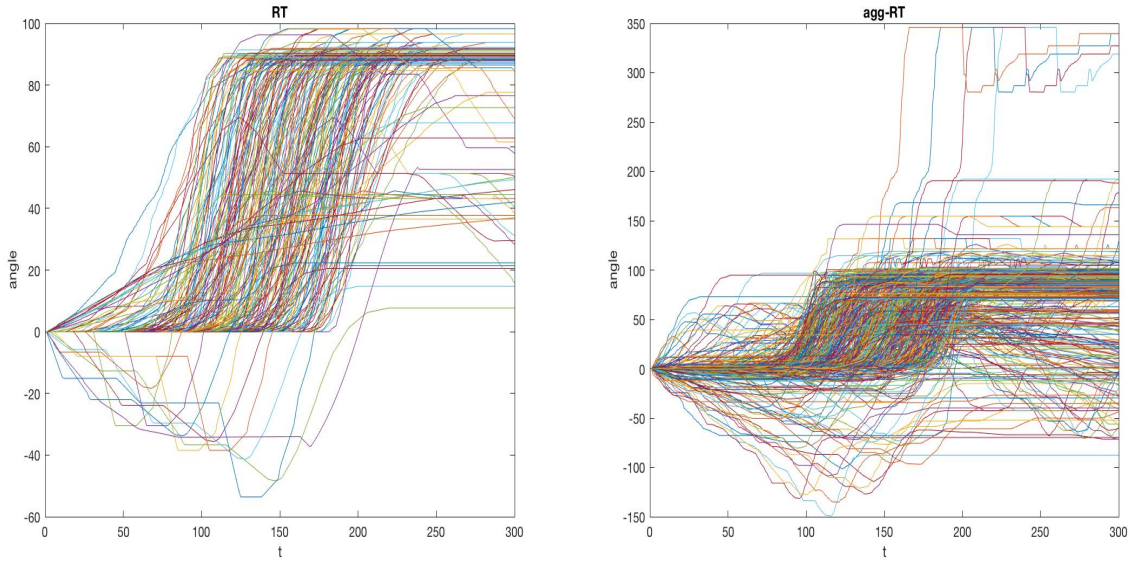


Figure 4.6: Vehicle Angle of Right Turn Sequences and Aggressive Right Turn Sequences from the Dataset

Plots show time vs angle data for right turning manoeuvres. In an aggressive right turn, most of the angles are more sharply changing and waving and also event time (turn time) is smaller than the non-aggressive one in general.

4.3.2 Real-World Driving Behaviour Learning Dataset (DBL)

Our second dataset comes from a study we conducted over 2019-2022 to observe human driving behaviour in real-world conditions across a range of road types, driver age groups and experiences. The study gathered 50 volunteer participants who drove a predetermined route (See Figure 4.8) around the region of Waterloo, Ontario, Canada. Due to the COVID-19 pandemic, most drivers ended up driving the route entirely on their own, following directions from an onboard GPS navigation system. The study uses a modified 2010 Cadillac SRX (in Figure 4.8) which is equipped with a top-mounted Lidar, a front and two rear-facing radar, camera sensors and onboard computers and various readout displays for the passenger. The onboard computers collected GPS, map, Lidar, Video System (VLT) and CAN-BUS data. these sensor data, self-evaluation with exit survey and annotation data from the DBL project will be utilized to detect aggressive driver behaviour.

Data was collected on multiple road types (e.g. major highways, city centers and rural highways), under different traffic conditions (e.g. rush hour, traffic congestion times, low traffic) and under different environmental conditions (winter, spring, summer). The same route was repeated twice for each participant, with a short break in between. The route length is around 57 km which takes about 1 hour and 10 min to complete without traffic. One tour consists of approximately 12 Right turns, 11 Left turns, one forced left lane change, three forced right lane changes and two straight roundabouts. These numbers are doubled for the second trip of each participant. There is an exit survey to calculate the potential aggressiveness score for each participant. Each question's answers have an aggressiveness score based on the potential of aggressiveness and the sum of all questions' answers gives an initial score of the participant driver. This total score helps to label some events which are not able to apply thresholding. Figure 5.5 shows two samples of route location and speed plots, plot at the top illustrates the position and speed data driven more aggressively than the bottom one since the driver's initial aggressiveness score, calculated based on the exit survey question, at the top plot is higher than the driver's initial aggressiveness score at the bottom plot.

Route

City roads (urban environment) trips are compromised of city-specific features such as: pedestrians, crosswalks, traffic lights, stop signs, cyclists, roads affected by building constructions, parked cars, turns, narrow roads, etc. The highway parts consist of features such as: curved roads, multi-lane sections, on and off-ramps, connections between highways via on-ramps, varying speed limits, and highway divisions. The country roads include fea-

tures such as: stop signs, yield signs, roundabouts, traffic lights, single-lane two-way roads, turns, varying speed limits, traffic, etc.

Participants and Procedure

All participants were selected as experienced drivers with full 'G' licenses to drive in Ontario on all road types. Participants were prioritized based on the diversity of the applicant's information to ensure that an appropriate cross-section of the population were selected. An annotator sat beside the participant for the first 24 participant drives. The first 20 participants' drives are labelled and utilized for driver behaviour analysis in this study.

Pre-processing

At first, rolling window in different sizes were applied to features to create a time series dataset in order to label and classify. Unlike simulated data, real-driven data does not have some initial parameters to label driver behaviours. Therefore two methods are used to label datasets for two different tasks: automatic labelling and manual labelling. To label automatically, Map data, Lidar data and annotator notes are used. After windowing, the events will be labelled according to these features by applying thresholding shown in Figure 4.10. These thresholds are selected based on the previous studies in [96, 9, 97, 98, 99]. If a window consists of two or more events, the majority vote will be applied to the label window. Each window will have just one label corresponding to the event class. There are different types of label classes based on manoeuvre types and behaviour types. Manoeuvre labels for automatic labelling are: right and left turns, acceleration, deceleration, coasting and stopped. These actions can be labelled automatically aggressive or normal based on thresholds.

In addition to automatic labelling, this dataset has been labelled manually by the investigator using the Matlab video labelling tool in Figure 4.9. Go-pro videos for each participant are watched through the app and labelled based on visual observation. The video and time-series data and automatically labelled labels are checked at the same time by the investigator in order to prevent biased decisions. Time series data and video are overlapped by setting the start location. Python Orange3 data visualization tool is used to double-check the labels with data features. More than 12 maneuver labels are defined in a manual labelling tool, if an aggressive action comes across, it was labelled as aggressive others are labelled as normal behaviours. A rolling time window is applied to create a dataset as input of the classification of both tasks. The dataset includes just labelled time windows, others are eliminated from the dataset.



Figure 4.10: Automated Data Labelling by Thresholding and Scoring Flowchart

4.3.3 Convolutional Neural Network Implementation

Convolutional neural networks consist of alternating convolution and pooling layers which allow more efficient learning of image data. CNN is a fully connected network by leveraging weight sharing and a locality assumption. One of the motivations for our research is to determine if we can use the efficient weight-sharing and locality assumptions of CNNs to develop a richer model for time series data.

To apply CNNs to the time series data, we first need to create a 2D dataset from the raw sensor data time series. We create 2D data making each feature such as position, velocity and angular velocity to define rows of the matrix used as input to the CNN. We then generate the columns of our data matrix using a sliding window across time on the original time series data. Following the windowing, the data is of dimension `windowsize`×`featuredimension`. This dataset is then input into the CNN for training.

We first employ a CNN for learning driving styles from the features derived by applying sliding windows to position, velocity and angular velocity features derived from raw time series. We create 2-D convolutions to apply on inputs. In our problem, a CNN is utilized to learn two different tasks which are manoeuvres classification and aggressive and normal driving behaviours classification. The allow translation invariance for driving patterns over time, max pooling layers are used. An illustration of the applied CNN classification layers architecture can be seen shown in Figure 4.11. This is a basic CNN classification which has two convolution layers. We apply this model to classify manoeuvres and aggressive and normal driver behaviours separately. In other words, four CNNs are created for four different purposes. First of all, a CNN is implemented to classify turn manoeuvres (right or left). Then a second CNN is used for lane-changing classification (right or left). After that, another CNN is built to classify manoeuvres (including both left and right turns, and right and left lane changes). Finally, a fourth CNN is trained for aggressive driver behaviour classification (aggressive or safe). These basic CNN classification models are the first step to see how the CNN performs on our problem.

For classification sliding window sizes of 200 and 300 steps are applied with step size of 2 on position, velocity and angular velocity data derived from SUMO **simulated data**. Each window is labelled based on the consistency of an event. Windows without any labelled events are labelled as 'other'. Turn events crossing window divisions were separated from the dataset and utilized for turn classification. The whole dataset consists of 163,089 events (turns and lane changes) in total. To achieve a class-balanced, the class with the fewest samples determined the train/test data set sizes (i.e. Aggressive Rlch has 7,625 samples). For turn event classification, 15,250 events, equally distributed between aggressive and normal, were used with 80% used for training and 20% for testing. Also, lane changes were

separated from the dataset and utilized for lane change classification. There are 15,250 lane changes which have been selected from the entire dataset consisting of half of them aggressive and the same percentages (80%-20%) of lane change events used for training and testing.

As with the simulated data, sliding window sizes of 2-3 seconds are applied with overlapping 2-step size on position, velocity, and angular velocity data coming from **DBL**. If the sample window has both manoeuvre and behaviour labels, it is selected as the main dataset to be used for the proposed classification method. The whole dataset has more than 120,000 sample event windows. However, in order to balance the datasets for each class, downsampling is applied and the aggressive left lane change class sample size is selected as the maximum number for each class which is 3,904. Samples from other manoeuvre classes are selected randomly based on this sample size. 15,616 event combinations have been generated for 10 runs including 7,808 aggressive actions. 70%, 10% and 20% are the percentages that are used for training, validation and testing respectively.

The implemented CNN has seven layers consisting of two 2D convolutions followed by a max pooling layer with size of two, fully connected layers and softmax layers. ReLu activation function, categorical cross entropy loss function and SGD optimizer are used and 0.0001 is selected as the learning rate. Zero-padding is selected to keep the variable length in the time series since the size of the feature maps would reduce at each layer without padding. Zero-padding allows you to build deeper networks.

4.3.4 Parallel Convolutional Neural Network Implementation

The schema of our **Parallel CNN (PCNN)** is illustrated in Figure 4.12. The Parallel CNN architecture we build for the aggressive driver identification problem consists of two CNN classifiers being trained and working in parallel as follows. The first Convolution and Pooling layers are used for both classification tasks in order to share weight and parameters since we propose that these two classification tasks are related and can support each other by sharing. The remaining convolution and pooling layers and fully-connected layers are separated to focus on their specific purposes for the classification problem. PCNN layers' input dimensions, kernel and pooling window sizes change based on the input data format. These can be found in the Tables 4.3,4.4 for both plain or rotated and converted time series datasets. By applying this method, efficiency for both results and time are improved. Before training this method on the dataset, we retrained previously built CNNs by adding new information to the input dataset in order to prove our expectations for the proposed method. We arranged manoeuvres labels to add on input of aggressive

and normal behaviour CNN classification as a new four-dimensional feature group. Also, aggressive and normal behaviour classes were used as an additional f-dimensional feature set for previous CNN manoeuvre classification.

In order to compare with PCNN, we implemented LSTM and HMM methods by using same plain time series dataset and features. For HMM, each class from eight classes has its own HMM. I trained eight separate HMM models with and the HMM classifier is working based on winning log-likelihood output from HMMs. Different numbers of hidden states are tested and three states are selected for this problem. Moreover, the same dataset is utilized to run LSTM consisting of bi-LSTM, fully connected and softmax layers. Adam is selected as the optimizer algorithm. Also, training data is shuffled for every epoch selecting random order and splitting data into batches.

Layers	**Kernel Size**	**Stride**	**Dimension**
conv1	[2,2]	[1,1]	300×4
pool1	[2,1]	[2,1]	150×4
conv2^1	[2,2]	[1,1]	150×4
pool2^1	[2,1]	[2,1]	75×4
conv2^2	[2,2]	[1,1]	150×4
pool2^2	[2,1]	[2,1]	75×4

Table 4.3: PCNN Layers Windows and Output Dimensions for Plain or Rotated Time Series Data Input

(300×4) =>($Time\text{-}window$ $\times$[x$_{position}$, $y_{position}$, $heading\text{-}angle$ $speed$]).

Layers	**Kernel Size**	**Stride**	**Dimension**
conv1	[3,3]	[1,1]	$3 \times 150 \times 150$
pool1	[2,2]	[1,2]	$2 \times 60 \times 120$
conv2^1	[3,3]	[1,1]	$2 \times 60 \times 120$
pool2^1	[2,2]	[1,2]	$1 \times 30 \times 120$
conv2^2	[3,3]	[1,1]	$2 \times 60 \times 120$
pool2^2	[2,2]	[1,2]	$1 \times 30 \times 120$

Table 4.4: PCNN Layers Windows and Output Dimensions for Converted Image Data Input: $(3 \times 150 \times 150)$

The proposed model was then applied to the selected 20 participants from the DBL

Aggresiveness rules	Threshold
acc/decc quickly(m/s^2)	|ax|>3
turn quickly(rad/s)	|gyroz| >1.2
speeding(km/h)	> speed limit+25
close following (m)	stand still distance<0.4
close following(sec)	head way time<0.5
frequent lane change($/min$)	total numbers of lane change>8

Table 4.5: Aggressive Driver Behaviours Determination Rules and Thresholds

dataset. Position, velocity and angle information that are coming from data are re-sampled and labelled manually by watching a driving Go-Pro video by a researcher of the project. We have **determination rules for aggressiveness** which are defined by previous studies [9],[97]. These rules which we considered to label the action as aggressive are speeding, accelerating and decelerating quickly, close following, frequent and improper lane changing, failure to signal and failure to obey traffic signals, racing, and frequently honking. For each action, start points are labelled manually by using the Matlab video labeller tool shown in Figure 4.9. Each participant's video is watched and labelled by the researcher and controlled these labels after each labelling process is done. Then, the annotator's notes were checked before and after the labelling process in order to catch some aggressive drivers' behaviours that can not be seen from the video such as shouting or honking. After visual labelling, the actions are also controlled and labelled by automatically considering aggressiveness rules which are defined numerically in Table 4.5.

The same range of window sizes was used to extract a time series of data points using a sliding window approach. After windowing, these windows are labelled according to the consistency of aggressive behaviour (normal/safe, aggressive) or manoeuvre labels (right turn, left turn, right lane change, left lane change) within these time windows.

4.3.5 Results

We first implemented four separate 2D CNN models (illustrated in Figure 4.11) for different tasks to apply on time series data for both simulated and real-world driven data. Turn classification results with a single task CNN as shown in Table 4.6 are 90.2% and 80.3% for training and testing on the simulated dataset. Lane changing classification with the single task CNN gives a little less than turning but still has captured significant features from the simulated dataset to classify. When these turn and lane-change events are all combined into

Classification Task	Train accuracy (%) on Simulated Dataset	Test accuracy(%) on Simulated Dataset	Train accuracy(%) on Real Driven Dataset	Test accuracy(%) on Real Driven Dataset
Turns Classification	90.2	80.3	92.1	90.5
Lane-changes Classification	88.2	75.6	80.6	72.7
Behavior Classification	65.6	60.8	80.5	78.4
Manoeuvre Classification	82.9	65.0	75.3	71.1
Behavior Classification with Manoeuvres label added to input features	73.1	65.0	88.2	82.7
Manoeuvre Classification with Behaviour label added to input features	89.2	70.3	78.2	75.0

Table 4.6: Accuracy Results for Different Tasks Using just the Single Task CNN on Time Series Datasets.

manoeuvres for another single task CNN classification, the accuracy decreases to 82.9% and 65.0% for training and testing on the simulated dataset. Since more classes increase the complexity of the problem, the combination of lane-changing and turn classification accuracy is lower than only lane-changes or turn classification accuracy. Aggressive and safe behaviour classification performance are 65.6% in training and 60.8% in testing because this CNN classification is more challenging task compared to others. When we include all 8 classes together, this classification task even becomes even more complex and the accuracy decreases to 70% and 61%, training and testing respectively, see Table 4.7. When single-task CNN models are fed with labels of other classification classes, classification accuracies rise up. For example, if aggressiveness classification samples include manoeuvre labels, the classification efficiency shows a remarkable increase from 80.5% to 88.2% in training accuracy on the real-world driven dataset. The results illustrate that the applied CNN classifications can improve their performance by adding other task labels as a feature set on input. We can infer from these results, PCNN can serve our purpose which is to show that high-level features derived from different tasks can support each other.

The PCNN model has been applied on **simulated time series** in order to classify these 8 classes. PCNN has improved accuracy results from 60.6% (single CNN model accuracy result) to 65.7% which suggests that sharing the early layers helps to improve accuracy in general. The comparison of the accuracy results table illustrates that the LSTM method is not working as well as PCNN in terms of general accuracy, but when investigating from the confusion matrix, for two classes (normal right turn and aggressive right lane-change), we find that LSTM works better than PCNN. Also, LSTM takes longer to train than PCNN. Similarly, while HMM does not work to detect aggressive manoeuvres classes

(general accuracy is approximately 20%), it gives the highest single accuracy for normal left and right turns classification among CNN, PCNN, and LSTM. However, we found that HMM results were very sensitive to any small changes in parameters. In general, CNN and PCNN are more efficient methods to detect essential features from aggressive behaviours comparing to the other methods considering both accuracy and cost. Moreover, a different feature engineering technique is applied on the simulated time series, that is angle of rotation to x and y locations. PCNN is applied on time series data which consists of these rotated locations, speed, and angle. This leads to significant improvement in results. Considering all 8 classes including both tasks, test accuracy jumps from 65.7% to 95% when position data rotated to a determined quadrant. The PCNN classification on different versions of dataset F-score results is shown in Table 4.8. These f-score results are means of validated model by training PCNN model 10 times on the simulated dataset. For each single training, randomly 20% of selected for testing, from the training set 20% has been selected for validation. The early stopping point is determined as 11 epochs and 100 sample sequences has been used for an epoch. Datasets which are applied of different window sizes (300 and 200) has also been compared in this table. In order to prevent overfitting, the best epoch number for training PCNN can be selected between 10 and 15 based on accuracy and loss results which are shown in Figure 4.13.

After that, the PCNN model is performed on time series data from DBL. Location, speed and heading angle information are utilized as input. While CNN shows great performance in accuracy for a single task, PCNN gives significant results in accuracy for multi tasks on DBL. These results indicate that CNN has the ability to gather high-level information from time series efficiently. PCNN multitask classification model's accuracy/loss vs epoch plot in Figure 4.14 illustrates that 99% accuracy in behaviour classification, 86% accuracy in manoeuvre classification for 3 sec window size. Stopping epoch numbers can be selected from between 40 and 60 based on these plots as well to prevent overfitting. The PCNN classification on different versions of DBL dataset F-score results is shown in Table 4.8. These f-score results are means of the validated model by training 10 times the PCNN model on the dataset with different random seeds.

4.4 Conclusion

The contributions of the study in this chapter are: (1) Introduction of automated data production by the SUMO and Webots simulators for a set of random road configurations and various driver behaviours, (2) Real-world driving dataset application, (3) Applying CNNs on time series for driver behaviour analysis, and (4) Introduction of a parallel CNN

Method	Train accuracy(%)	Test accuracy(%)
CNN	69.5	60.6
PCNN	71.8	65.7
LSTM	65.8	58.13
HMM	55.28	51.22
SVM	63.4	60.5

Table 4.7: General Accuracy Results of Classification for 8 Classes Together with Various Models Applied on Whole **Plain Simulated Time Series Dataset**

Method	**Manoeuvre Classification** F1 score	**Behaviour Classification** F1 score	**Manoeuvre Classification** F1 score	**Behaviour Classification** F1 score
Time window size	3sec	3sec	2sec	2sec
PCNN Simulated Time Series Data	0.75	0.65	0.72	0.51
PCNN on Simulated Time Series Data with Rotated x and y	**0.97**	0.94	**0.96**	0.90
PCNN on DBL Time Series Data	0.87	**0.98**	0.85	**0.95**

Table 4.8: PCNN Application Results Comparison Table on the Different Type of Simulated data

architecture with a shared early stage and split later stages for dual training of behaviour and manoeuvre classifiers. We proposed Parallel Convolutional Neural Networks to classify aggressive driver behaviours together with manoeuvre classification. Two CNNs in the proposed method were trained and tested separately on large simulated and real-world driving datasets. These experiments show that these CNN models are able to capture high-level features to differentiate driving manoeuvres and aggressive driver behaviours. Moreover, the results illustrate that CNN classifications can work in parallel and can support each other efficiently. This initial step was important for designing PCNN to build driving assist systems using embedded smartphone sensors.

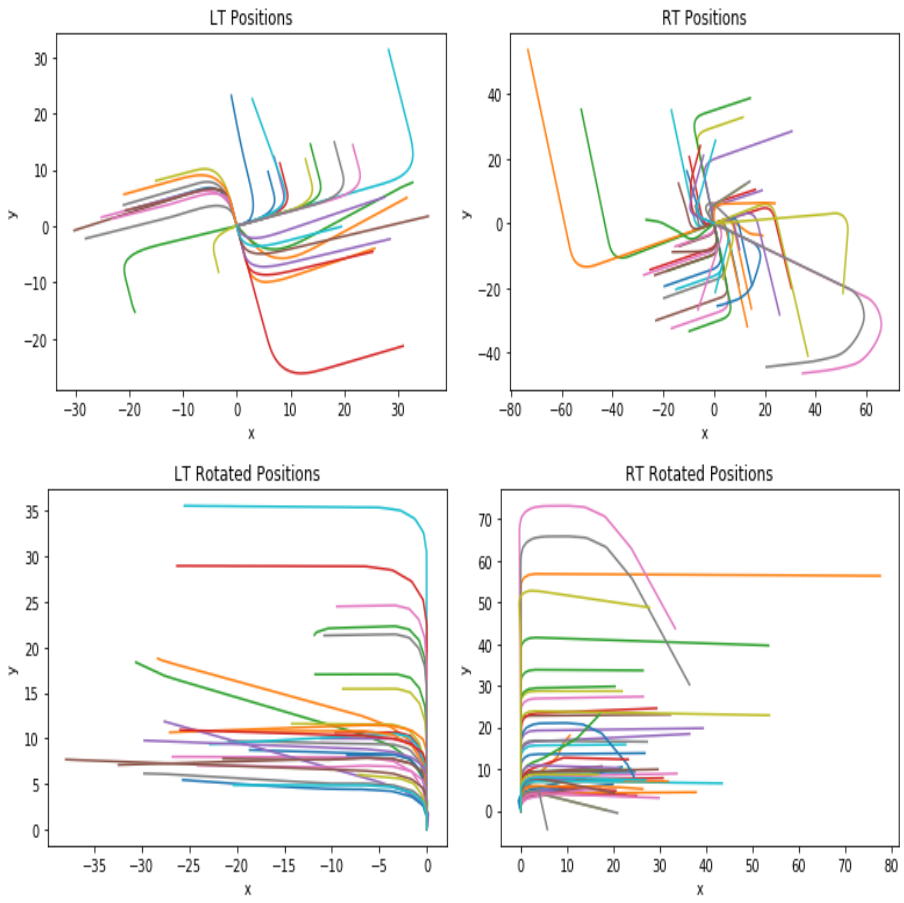


Figure 4.5: Zero Centered Vehicle X and Y Position Data Samples for 300 Time Window from the Dataset

Plotting at the top show left and right turning manoeuvre's x and y location data from the Simulator, plotting at the bottom illustrates rotated vehicle x and y position data samples for 300 time window from the Dataset.

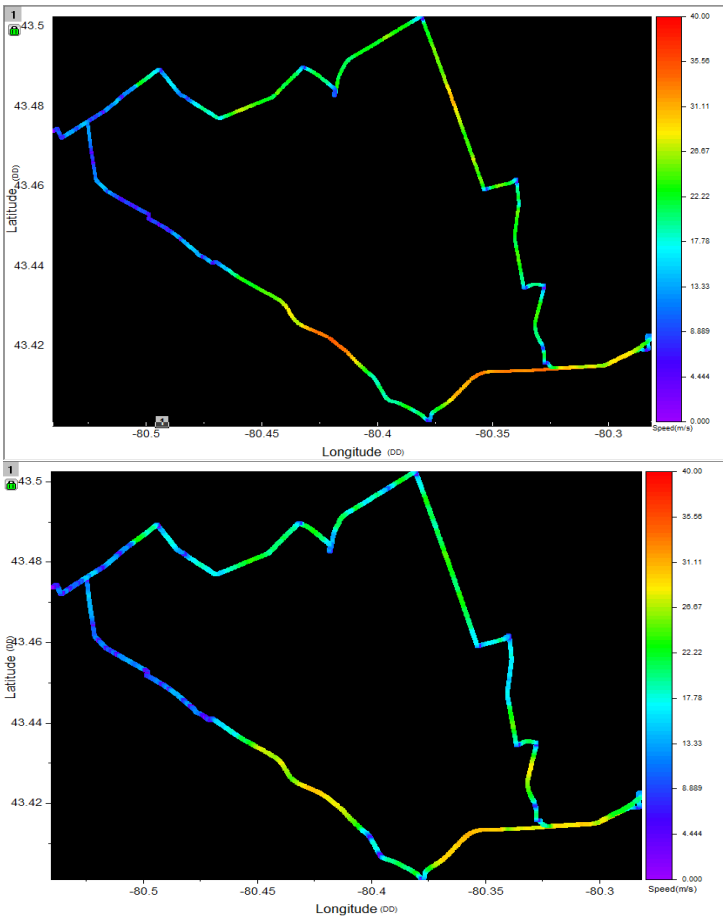


Figure 4.7: Two Different Participants' Route-speed Plot

at the top is a sample of a more aggressively driven trip, at the bottom is a sample of a driver with normal behaviour

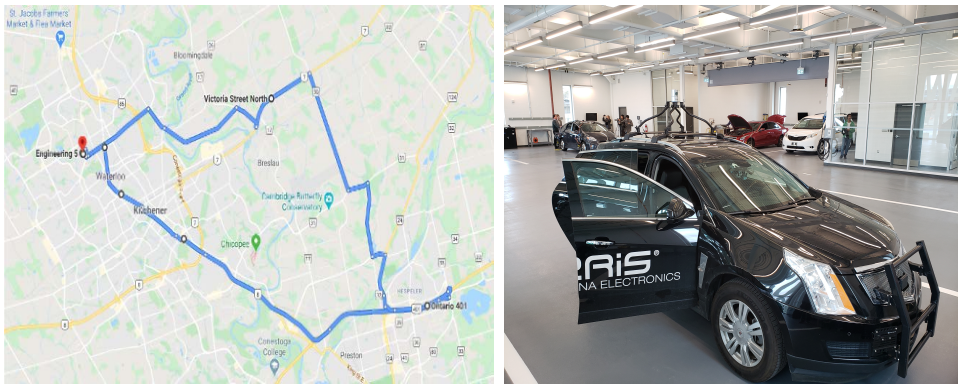


Figure 4.8: The Route that Is Used for Participant Drives

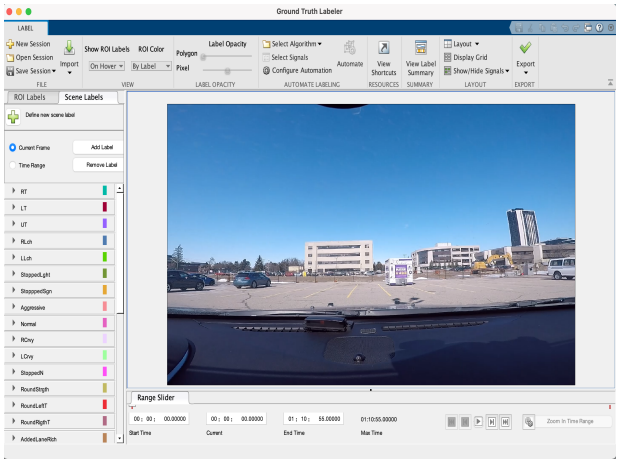


Figure 4.9: Manual Labelling Tool

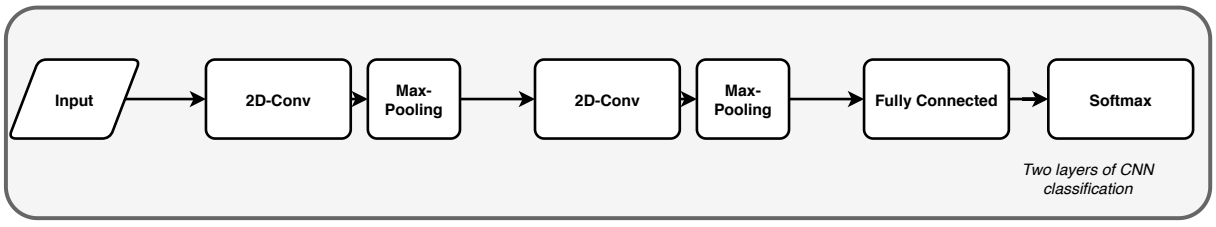


Figure 4.11: 2D-CNN Architecture

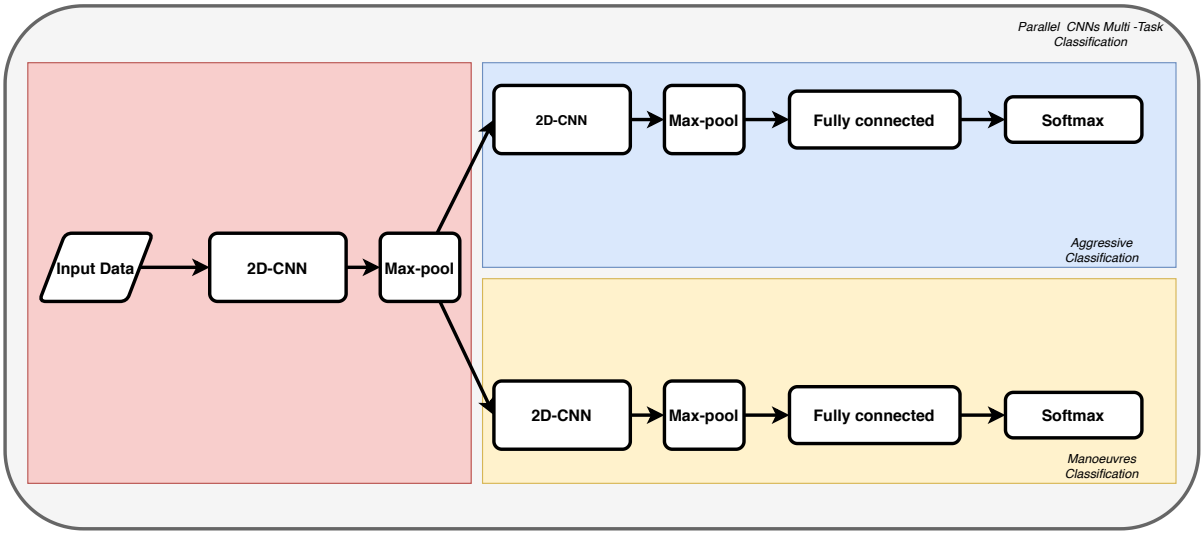


Figure 4.12: Parallel CNN Architecture for Two Different Tasks, Driving Manoeuvres and Driver Behaviors

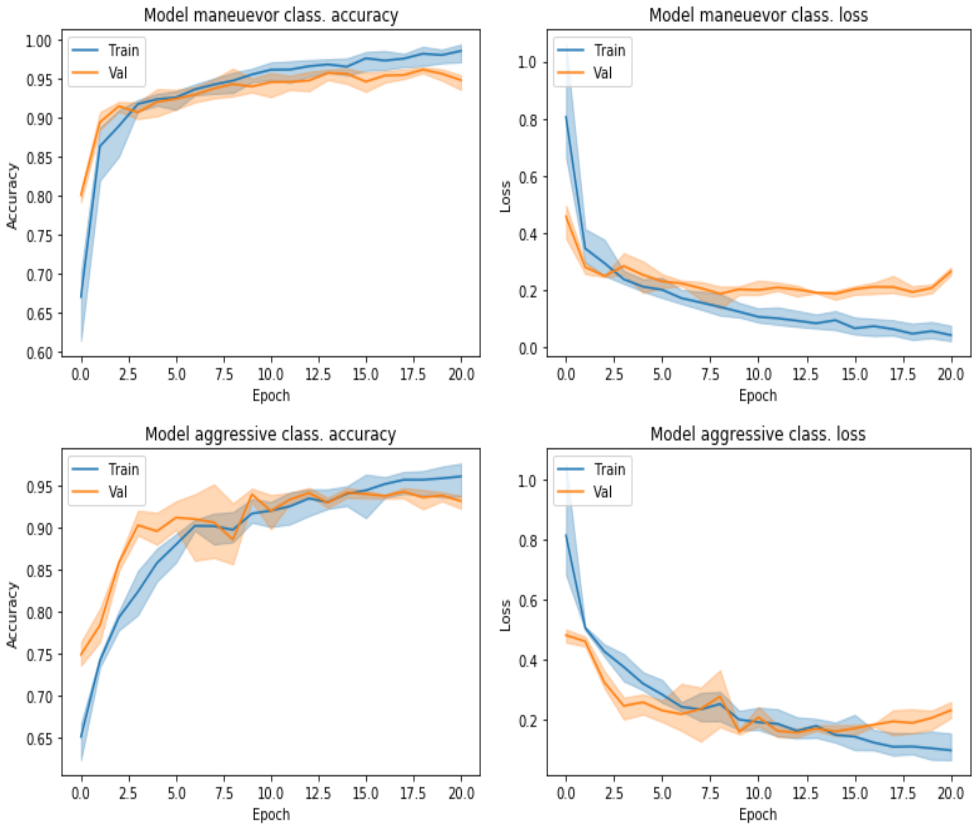


Figure 4.13: PCNN Classification on **Rotated Time Series of Simulated Data** Accuracy and Loss vs Epochs plotting:

The figures at the top show manoeuvres classification accuracy and loss plotting in PCNN, The figures at the bottom illustrate aggressive and normal behaviour classification accuracy and loss plots in PCNN. 10 times training runs were used for these plots.

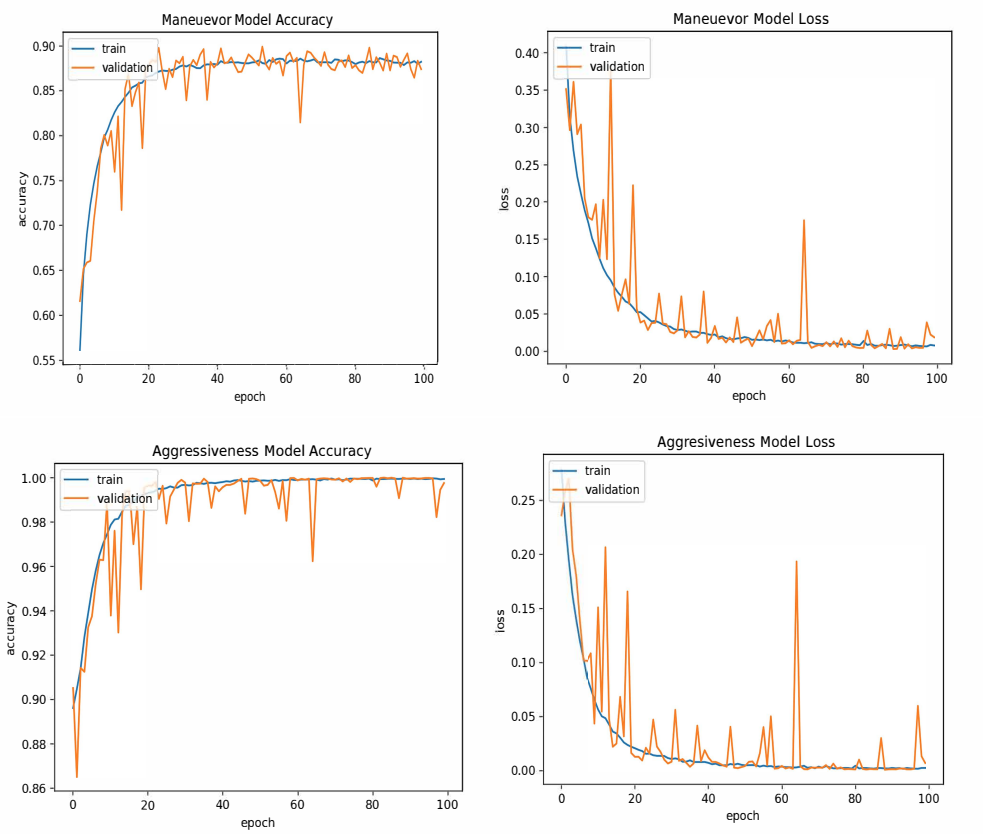


Figure 4.14: PCNN Classification on **Real Driven Dataset** Accuracy and Loss vs Epochs plotting

The figures at the top show manoeuvres classification accuracy and loss plotting in PCNN, The figures at the bottom illustrate aggressive and normal behaviour classification accuracy and loss plots in PCNN.

Chapter 5

Spatial PCNN Classification for Aggressive Driver Behavior with Attention layer

5.1 Introduction

The neural network-based method can omit the process of handcrafted extracting features, which can greatly reduce the workload of feature engineering, and improve the recognition accuracy. Inspired by the success of deep learning, many neural networks, including convolutional neural networks (CNN) and recurrent neural networks (RNN), have been developed to recognize sensor-based activity [100]. CNNs have had a significant impact on the field of image recognition and have many variants. Although CNNs are highly capable of extracting spatial features from raw data which is widely used in image classification [100], the existing CNN architecture does not have good spatial-temporal features. The spatial-temporal relationship is important in learning a time series. In particular, we propose PCNN in the previous section to classify aggressive and normal driver behaviours. In this chapter in order to eliminate handcrafted features, to get more spatial relationships and to increase the efficiency of classification, we propose a novel deep neural network for driver behaviour analysis by constructing a new PCNN model adding spatialCNN and attention layers. We highlight the main contributions of this work as follows: A new neural network structure, called Spatial Parallel CNN with Attention (SPCNN-A) is developed. The model is a novel method by using multi-tasks in parallel, by including spatialCNN with attention in terms of applying it on time series data. Spatial CNN and self-attention

have common usage in image processing rather than time series. The network is suitable for time series data with correlations between features (such as vehicle sensor data). The optimized network structure represents the correlation between the features. Time series application and construction of the model for multi-tasks in parallel contribute to this approach novelty. Our main contributions are:

- Introduction of real-world driven data with aggressive and maneuver driver features and labels.
- Deriving the impact of aggressive personal features by classification.
- Improving the developed parallel CNN architecture by using spatial CNNs and adding attention.
- Adapting the method to real-world driven data.

The rest of this chapter is organized as follows. The utilized tools in this work are reviewed in the following section. Then, experiments and data are covered, followed by a full description of the proposed SPCNN-A model. In the final section of this chapter, we concluded with the results.

5.2 Tools

5.2.1 Spatial Convolutional Neural Network (SCNN)

In order to learn the spatial relationship between features more efficiently, we use the Spatial Convolutional Neural Network (SCNN) method which is proposed in [1]. 'Spatial' does not mean the application of 'spatial convolution', but indicates *creating* spatial information by using a specific CNN structure. This structure is illustrated in the 'SCNN' module of Figure 5.1. SCNN is applied on a 3-D tensor ($C \cdot H \cdot W$), where C, H, and W means the number of channels, rows, and columns, respectively. The 3D tensor is divided into H slices. Then the first slice is input to a convolution layer with determined C and w (kernel numbers and kernel width respectively). The output of the CNN is appended to the next slice to create a new slice, and this new one is fed to the following convolution layer. This process continues until the last slice is fed to the CNN. $\mathbf{K}$ denotes a 3-D kernel tensor consisting of $K_{i;j;k}$ elements which are the weight between an element in channel i of the last slice and an element in channel j of the current slice, with an offset of k columns

between two elements. In addition, there is an input 3-D tensor $\mathbf{X}$ consisting $X_{i;j;k}$ elements, where i, j, and k indicate indexes of channel, row, and columns respectively. The forward computation of SCNN is:

$$X'_{i;j;k} = \begin{cases} X_{i;j;k}, & if\ j = 1 \\ X_{i;j;k} + f(\sum_m \sum_n X'_{m;j-1;k+n-1} \times K_{m;i;n}), & j = 2, 3, 4, ..., H \end{cases}$$

where f is a nonlinear activation function such as $ReLU$. The X' denotes the updated element. There is a recurrence since the convolutional kernel weights are shared by all slices.

There is a direction in a *SCNN*. In the basis model of *'SCNN'*, there are four submodules for downward, upward, rightward, and leftward directions which are represented with their suffix 'D','U','R','L' respectively. SCNN views rows or columns of feature maps as layers and extracts the feature from bottom to top, top to bottom, left to right and right to left according to the traditional CNN method. The extraction operation adopts a cascade mode. The input of the subsequent operation is the output of the preceding operation. In the traditional CNN structure, feature extraction is carried out on a layer-by-layer basis. The feature map extracted from the previous layer propagates to the next layer as the features of input extraction. For information interaction between pixels that are far apart, a larger convolution kernel or a deeper network is often needed to improve the receptive field of convolution kernel.

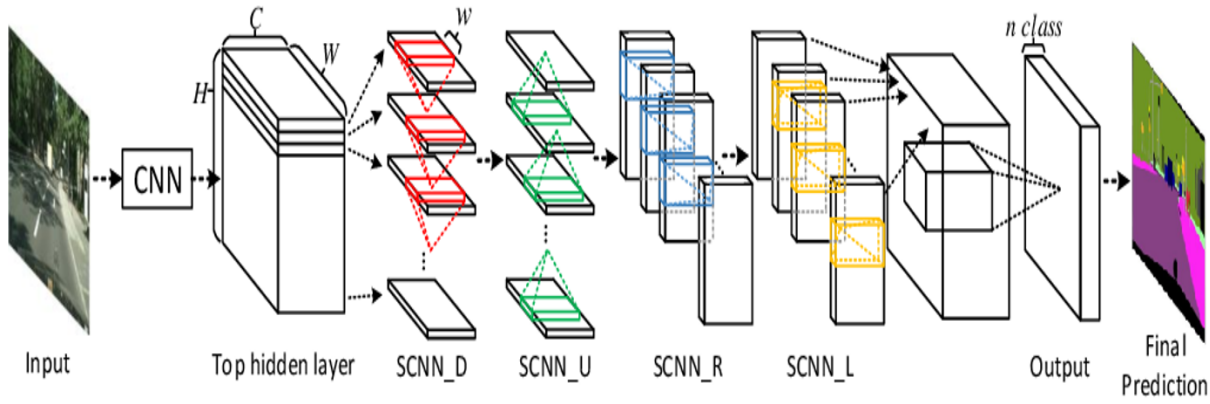


Figure 5.1: Spatial-CNN Architecture for Image Classification in [1]

5.2.2 Attention

Attention in CNNs can be categorized into two main classes that are hard (stochastic) and soft (deterministic). Hard attention selects a region of the image to attend at a time. It is effective in terms of calculation at inference time, but the model is non-differentiable and relies on complex techniques, e.g. variance reduction or reinforcement learning [101]. Therefore, this complicated technique makes optimizing these models more difficult. In the soft attention model, the weights are learned and placed 'softly' over all the regions from the image [102]. Unlike hard attention, this attention model is smooth and differentiable, but it is expensive with respect to the input size. Luong, et al. [101] proposed the 'global' and 'local' attention models. Global attention is similar to soft attention, while local attention is an interesting blend between hard and soft. An improvement over the hard attention to make it differentiable; the model first predicts a single aligned position for the current target word, and a window centred around the source position is then used to compute a context vector.

Attention is frequently used as a computational module to model sequences since it is an efficient model to capture long-distance interactions [102]. The first research in this area [102] was established by combining attention and an RNN [103] for alignment in Machine Translation. After that, the self-attentional transformer model was proposed as an extension of attention, and this model became state-of-the-art for Natural Language Processing tasks [104]. Integrating self-attention with convolution has recently been used in [105],[106] studies of language processing and reinforcement learning. Their architecture includes alternating convolution and self-attention layers. Attention has been widely used in visual tasks to alter problems due to using convolutions[107]. An attention mechanism can be used to weigh the importance of features from different spatial locations in an image. This allows the model to pay more attention to globally relevant or contextually important parts of the input, overcoming the inherent locality issue of standard convolutional layers. Most of these previously proposed attention methods over images [108, 109] mention the convolution operator's locality and weakness in identifying global contexts. These methods are proposed to detect long-range dependencies by re-weighting and refining convolutional feature maps.

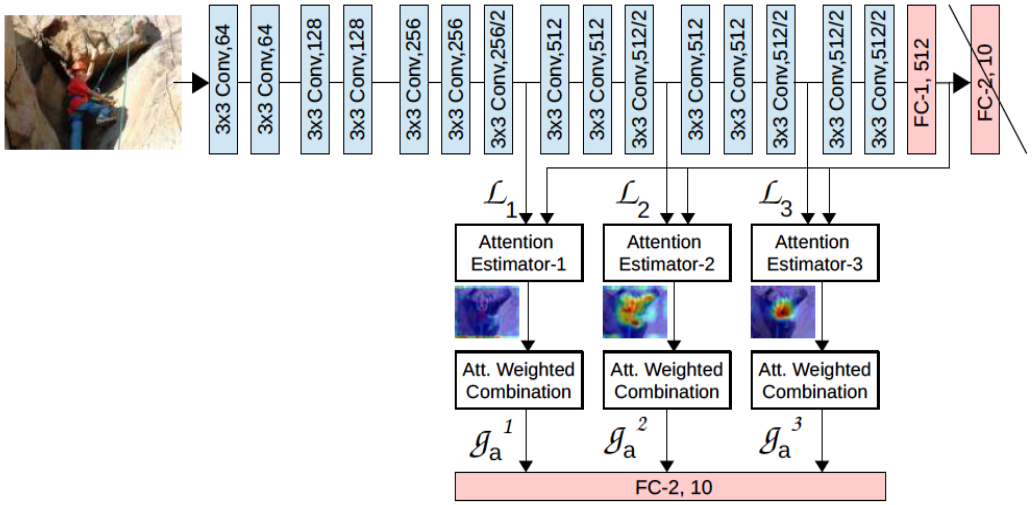


Figure 5.2: Attention-CNN Architecture for Image Classification in [2]

This approach is illustrated in Figure 5.2. $L_s = \{l_1^s, l_2^s, ..., l_n^s\}$ denotes the set of feature vectors coming from a given convolutional layer $s\epsilon\{1, ..., S\}$. Each l_i^s is the vector of output activations at i^{th} spatial location. There are n total spatial locations in the layer. g denotes the global feature vector having the entire input image as support. It is output by the network's series of convolutional and nonlinear layers with the assumption of the existence of a compatibility function C. This function calculates a scalar compatibility score by taking two vectors of equal dimension as arguments. The method first starts with computing, for each of one or more layers s, the set of compatibility scores $C(\hat{L}^s, \mathbf{g}) = \{c_1^s, c_2^s, ..., c_n^s\}$ where $\hat{L}^s$ is the image of L^s under a linear mapping of the l_i^s to the dimensionality of $\mathbf{g}$.

The compatibility score function C can be defined in different ways. The model from Bahdanau et al.[102]; Xu et al. [110], C is defined as follows:

$$c_i^s = \langle u, l_i^s + g \rangle, i\epsilon\{1...n\}(2)$$

It means concatenation of the two descriptors to an addition operation, without loss of generality. Another technique is to apply the dot product between g and l_i^s as a measure

of their compatibility:

$$c_i^s = \langle l_i^s, g \rangle, i\epsilon\{1...n\}(3)$$

The relative magnitude of the scores depends on the alignment between g and l_i^s in the high-dimensional feature space and the strength of activation of l_i^s.

Then, the softmax operation is used to normalize the compatibility scores:

$$a_i^s = \frac{exp(c_i^s)}{\sum_j^n c_j^s}, i\epsilon\{1...n\}$$

The simple element-wise, weighted averaging applied on the normalized compatibility scores $A_s = \{a_1^s, a_2^s, ..., a_n^s\}$ produces a single vector $g_a^s = \sum_{i=1}^n a_i^s.l_i^s$ for each layer s. Then, the global descriptor for the image $\mathbf{g}$ can be represented using the g_a^s. For a network trained under the restriction that g_a^s alone is used to classify the input image, A denotes the 'attention' map. In the case of a single-layer network ($S = 1$), the attention-incorporating global vector g_a is computed as described above, then transformed into a T-dimensional vector. This vector is subsequently passed through a softmax layer to derive the class prediction probabilities $\{\hat{p}_1, \hat{p}_2, ..., \hat{p}_T\}$, where T indicates the total number of target classes. For multi-layer networks ($S > 1$), there are two possibilities: 1) Concatenate the global vectors into a single vector $g_a = [g_1^a, g_2^a, ..., g_S^a]$ and use this for the linear classification step as in the single-layer case, or 2) employ S different linear classifiers and average the resulting class probabilities. All adjustable network parameters are learned via end-to-end training under a cross-entropy loss function.

5.3 Parallel Spatial Convolutional Neural Network implementation with Attention

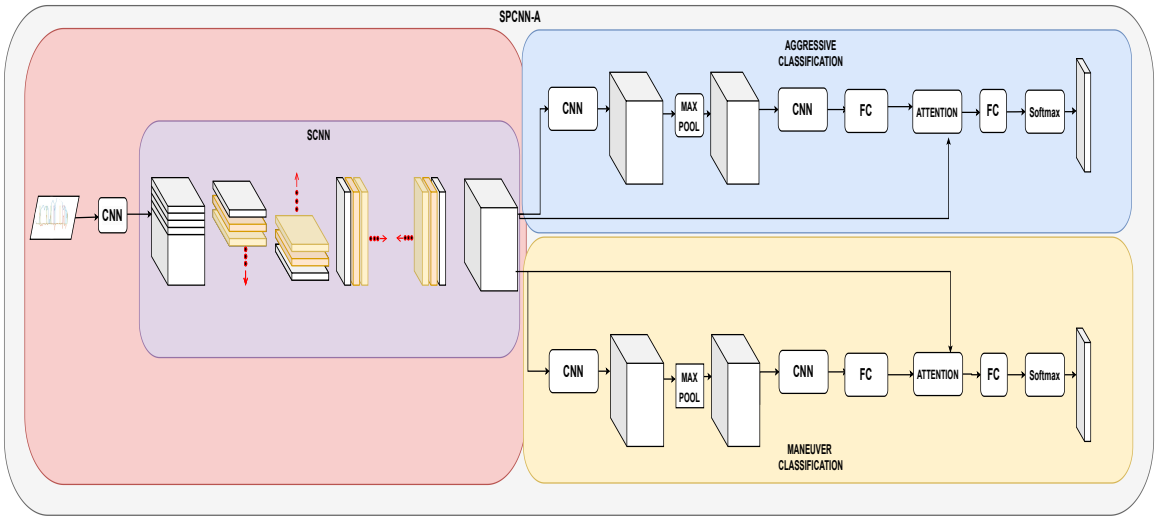


Figure 5.3: SPCNN with Attention Architecture for Driver Behaviour Classification

The architecture of Parallel Spatial CNN with Attention (PSCNN-A) is illustrated in Figure 5.3. The PSCNN-A model to improve PCNN model for the aggressive driver identification problem consists of two CNNs, a SCNN and attention layers being trained and working in parallel as follows. The first spatial convolution layer is used for both classification tasks in order to share weight and parameters since we propose that these two classification tasks are related and can support each other by sharing. Applying spatial CNN at an early stage is important to get more spatial features from time series and to deal with locality issues. The remaining convolution, attention, and fully-connected layers are separated to focus on their specific purposes for the classification problem. By applying the PCNN method in the previous chapter, we illustrate that utilizing two different tasks in parallel efficiently works in terms of accuracy and time. On the other hand, we realize that some of the spatial relations were missed by CNN while applying PCNN due to CNN's locality. We improved classification accuracy by adding spatial CNNN and attention to the PCNN method. Therefore, our two main purposes for creating this model are to catch more spatial relationships and capture more global features with local features from time series data. With these purposes, we additionally include spatial-CNN and attention layers

in the PCNN model. While spatial CNN can capture more spatial relationships between features, attention can solve locality and low-cost issues. Spatial-CNN is applied as the first layer and this CNN should only include convolution layers. We choose to omit max pooling since we do not want to miss any spatial relationships. Attention layers are placed at an early stage and late stage since local features are extracted at an earlier stage, while global features are derived at late stages in the parallel layers. Another reason why we have added attention layers is that we can experiment with a wider time window and attention can deal with a variety of event length time.

5.4 Experimental Analysis on the Real-world Driven DBL Dataset

In this chapter, we study an important real-world problem of driver behaviour analysis and propose another PCNN method consisting of spatial CNN and attention layers. The proposed solution has been implemented in vehicle behaviour classification which is shown in Figure 5.3. The experimental system has two main parts: data pre-processing and manual labelling shown in Figure 4.9, and a deep neural network model structure illustrated in Figure 5.3. To validate the proposed approach, a real-world driven dataset is utilized from the Driver Behaviour Learning (DBL) study data see Chapter 4 for a full description of the data preparation. In this study, the accelerometer, gyro meter, and GPS sensor information will be utilized as input of the model. This dataset has been labelled manually by the investigator using the Matlab video labelling tool in Figure 4.9. Go-Pro videos for each participant are watched through the app and labelled based on visual observation. The video and time-series data and automatically labelled labels are checked at the same time by the investigator in order to prevent biased decisions. Time series data and video overlap by setting the start location. The Python Orange3 data visualization tool shown in Figure 5.4 is used to double-check the labels with data features. More than 12 manoeuvre labels are defined in the manual labelling tool, if an aggressive action comes across, it is labelled as aggressive, and others are labelled as normal behaviours. The rolling time window is applied to create dataset as input for the classification of both tasks. Three manoeuvre class labels are added to four labels shown in Table 5.1 in this chapter for classification. A window size of 5 sec is used as sliding windows to separate the time series data into windows of uniform size for training. The dataset includes only labelled time windows, windows with no available labels are eliminated from the dataset. Figure 5.5 shows two samples of route location and speed plots, plot at the top illustrates the position and speed data driven more aggressively than the bottom one since the driver's initial

aggressiveness score, calculated based on the exit survey question, at the top plot is higher than the driver's initial aggressiveness score at the bottom plot.

AGGRESSIVE BEHAVIOUR LABELS	MANEUVER LABELS
AGGRESSIVE	RT
	LT
NORMAL	RLCH
	LLCH
	STOPPED
	MERGED HW
	EXIT HW

Table 5.1: List of Manoeuvre and Aggressive Behaviour Labels Used SPCNN-Att Classification Task

We use 80% of the dataset for training(70% for training and 10% for validation) the method and the other 20% are used to test. Before training, the dataset is balanced in terms of including a number of events and behaviours to prevent bias. Training is repeated for 100 times and calculated averages of the evaluation measurements, for each training and testing set are selected randomly. Each single training run has 60 epochs with a batch size of 50 sampled training windows per epoch.

5.5 Results

We first implemented the SPCNN model on the real-world driven data without attention and then added attention to that model to examine the effects of each additional layer. Turn and manoeuvre classification results are shown in Table 5.2. SPCNN outperforms PCNN on both tasks for both training and test sets in Table 5.2. Specifically, for manoeuvre classification, there is an improvement from 85.02% to 88.45% in training accuracy and from 83.15% to 86.33% in test accuracy. For aggressive behaviour classification, we observe an increase from 96.31% to 97.2% for training accuracy and from 95.24% to 95.03% for test accuracy. This suggests that the additional features or architecture in the SPCNN model provide a significant advantage over the simpler PCNN model. SPCNN-A further

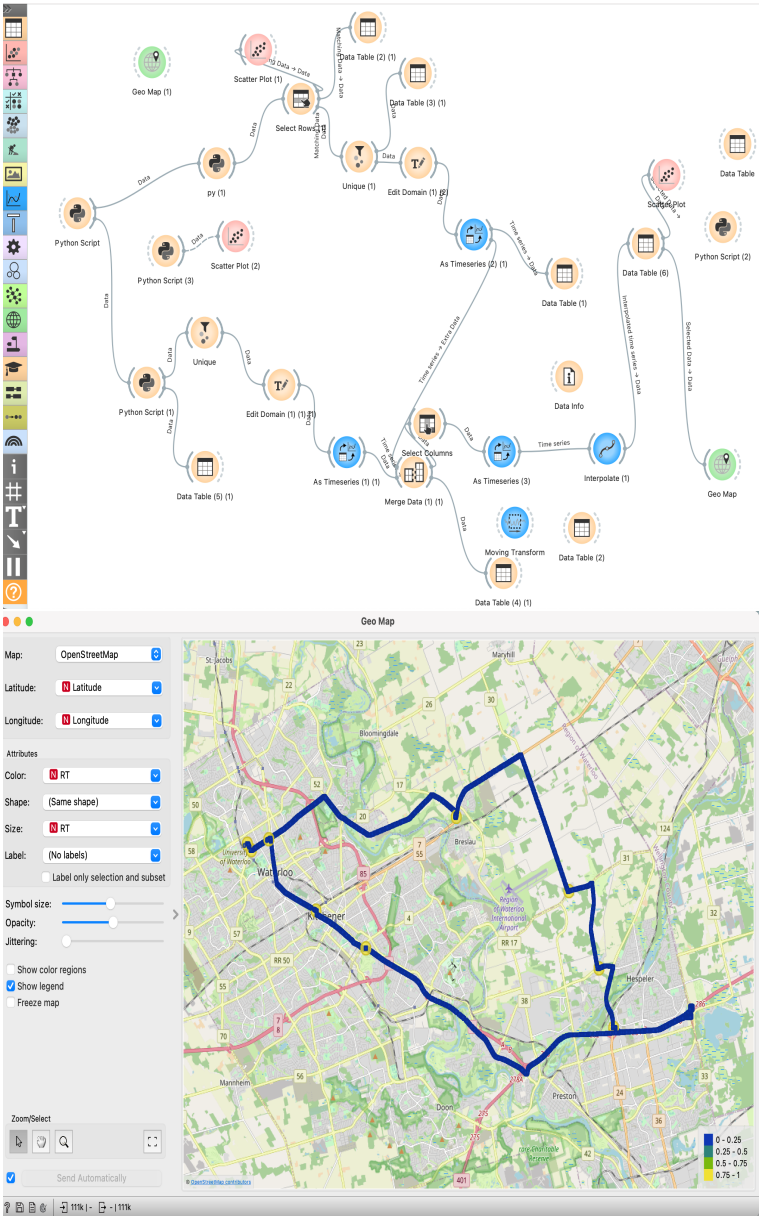


Figure 5.4: Orange3 Visualization Tool and Illustration of Latitude and Longitude Features with Turn Labels to Check Overlapping Time

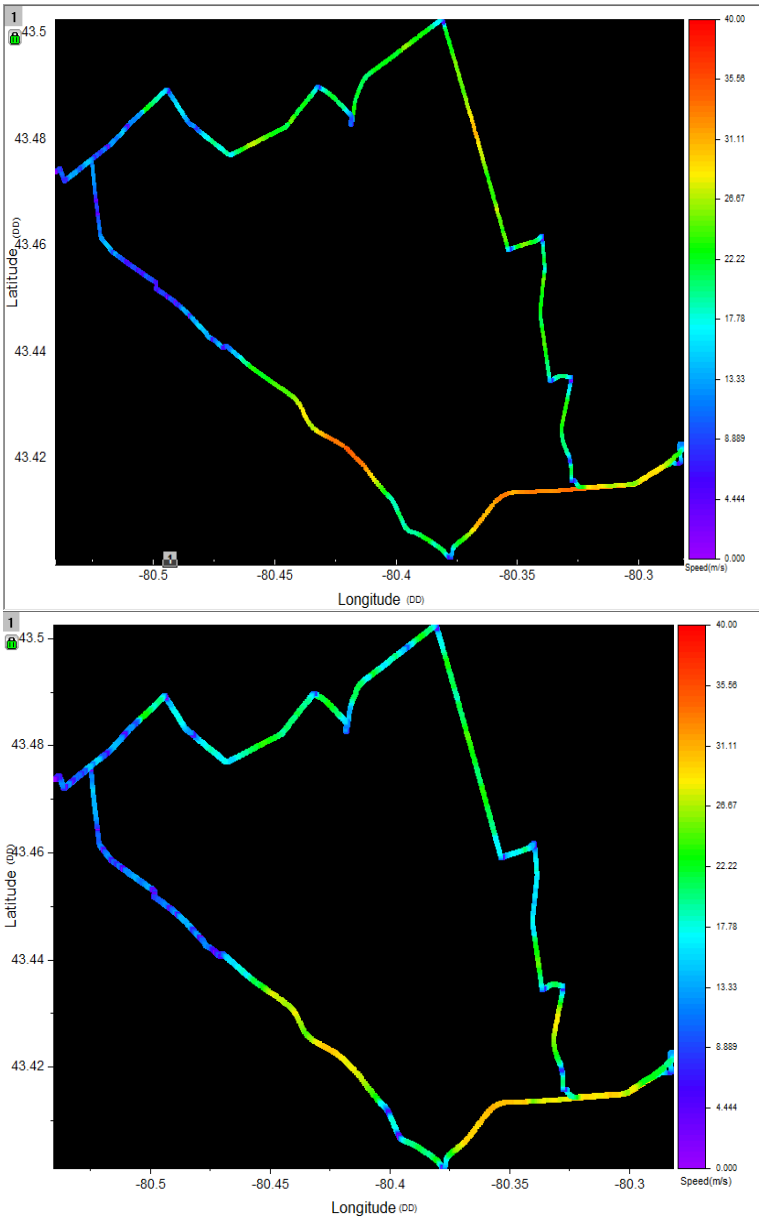


Figure 5.5: Two Different Participants' Route-speed Plot:

at the top is a sample of a more aggressively driven trip, at the bottom is a sample of a driver with normal behaviour

METHOD	TASK	TRAINING ACCURACY (%)	TEST ACCURACY (%)
PCNN	Maneuver Classification	85.02	83.15
	Aggressive Behavior Classification	96.31	95.24
SPCNN	Maneuver Classification	88.45	86.33
	Aggressive Behavior Classification	97.2	95.03
SPCNN-A	Manoeuvre Classification	91.01	87.05
	Aggressive Behaviour Classification	99.08	98.13

Table 5.2: Training and Test Accuracies for the PCNN, SPCNN, and SPCNN-Att Methods on Two Tasks

improves the performance of SPCNN, with the training accuracy for manoeuvre classification increasing from 88.45% to 89.01%, and the test accuracy increasing from 86.33% to 88.05%. For aggressive behaviour classification, the training accuracy jumps from 97.2% to 99.08%, and the test accuracy increases from 95.03% to 98.13%. These gains suggest that the attention mechanism in SPCNN-A is effective in refining the performance of the model. All models perform better on the aggressive behaviour classification task than the manoeuvre classification task, suggesting that the aggressive behaviour task might be less complex or more straightforward for the models to learn. For example, the SPCNN-A achieves 99.08% training accuracy and 98.13% test accuracy for aggressive behaviour classification, while for manoeuvre classification the numbers are lower at 91.01% and 87.05% respectively. The difference between training and test accuracies for all models on both tasks is small. This implies that all models generalize well from training data to unseen data, and overfitting is not a significant concern here.

SPCNN-A multitask classification model's accuracy/loss vs epoch plot at Figure 5.6 illustrates that 99.8% accuracy in behaviour classification, 91% accuracy in manoeuvre classification with 4 classes for 5-sec window size. Stopping epoch numbers can be selected from between 40 and 60 based on these plots as well. When we increase manoeuvre classes to 7, SPCNN-A multitask classification model's accuracy/loss vs epoch plot at Figure 5.7 illustrates that 99.7% accuracy in behaviour classification, 88.01% accuracy in manoeuvre classification with 4 classes for 5-sec window size. Stopping epoch numbers can be selected from between 40 and 60 based on these plots as well.

SPCNN-A significantly outperforms both LSTM and SVM in both manoeuvre and aggressive behaviour classifications in Table 5.3. When we applied these F1 and accuracy scores to paired Student-t test statistical analysis with the same sets, application of 5×2 cross-validation each time SPCNN-A method stronger than other compared methods with more than 95% confidence level in aggressiveness classification. The F1 scores are 0.85

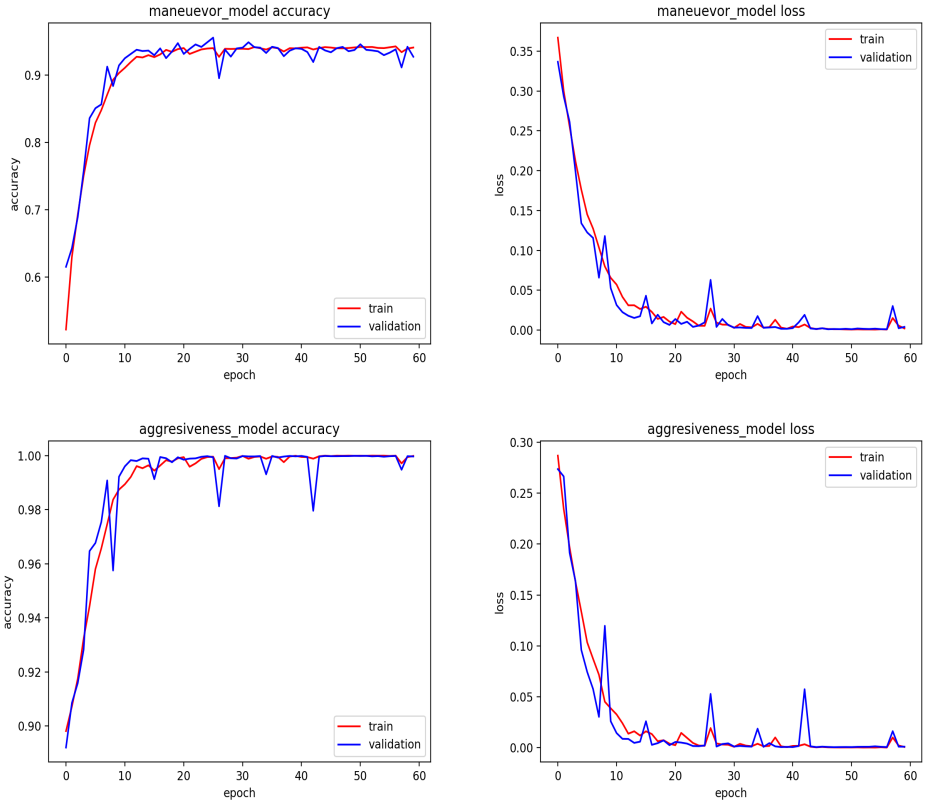


Figure 5.6: SPCNN-a Classification on DBL Dataset (for Four Manoeuvre Classes) Accuracy and Loss vs Epochs plotting

The figures at the top show manoeuvres classification accuracy and loss plotting in SPCNN-A, The figures at the bottom illustrate aggressive and normal behaviour classification accuracy and loss plots in PCNN.

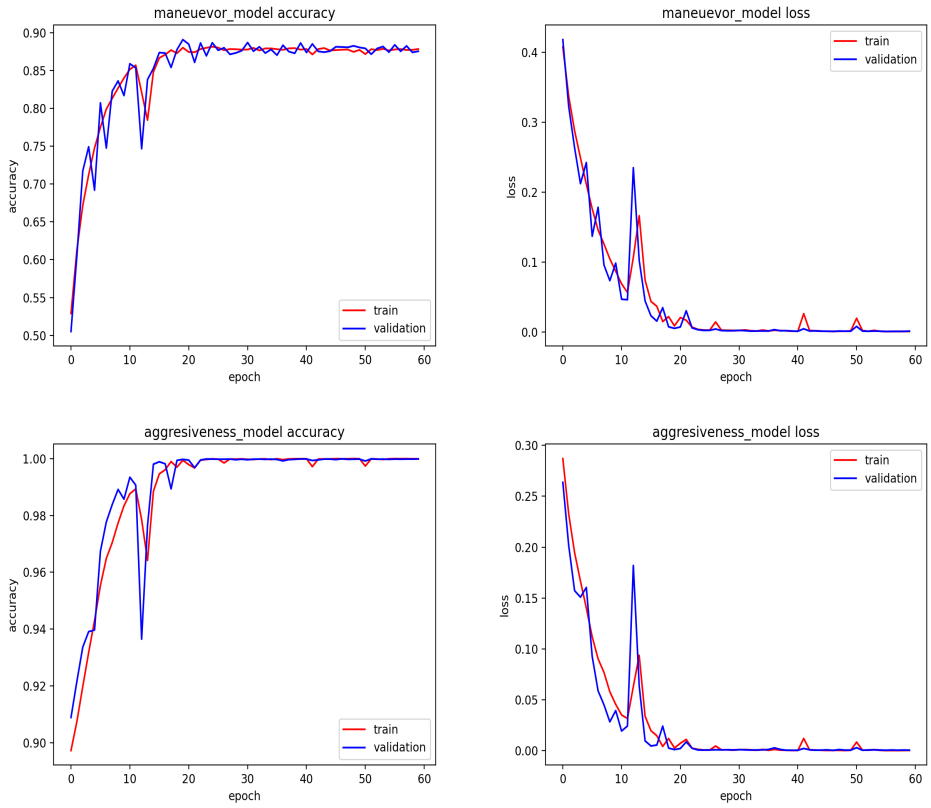


Figure 5.7: SPCNN-A Classification on DBL Dataset (for 7 Manoeuvre Classes) Accuracy and Loss vs Epochs Plotting

The Figures at the Top Show Manoeuvres Classification Accuracy and Loss Plotting for 7 Classes in SPCNN-A, The Figures at the Bottom Illustrate Aggressive and Normal Behaviour Classification Accuracy and Loss Plots in SPCNN-A.

and 0.96 for SPCNN-att, respectively, which indicates that the attention mechanism of SPCNN-A is highly effective. For aggressive behaviour classification, LSTM performs better than SVM with a score of 0.87 vs 0.8. However, SVM outperforms LSTM in the manoeuvre classification with a score of 0.67 vs 0.61. LSTM running time and cost are the highest among them. SVM run time is shorter when we compare others. All three methods perform better on aggressive behaviour classification than manoeuvre classification. This may suggest that for the aggressive behaviour classification task, the features used are more effective.

Method	**Task**	**F1 Score**
LSTM	Maneuver Classification	0.61
LSTM	Aggressive Behavior Classification	0.87
SVM	Maneuver Classification	0.67
SVM	Aggressive Behavior Classification	0.8
SPCNN-A	Maneuver Classification	0.85
SPCNN-A	Aggressive Behavior Classification	0.96

Table 5.3: F1 Scores for Different Methods and Tasks

In general, SPCNN-A and PCNN are more efficient methods to detect essential features from aggressive behaviours comparing to the other methods considering both accuracy and cost.

5.6 Conclusion

In this chapter, we proposed a novel deep learning architecture for capturing more spatial relationships, combining local and global features and multi-task training. SPCNN-A, which combines spatial, convolutional and parallel components, classifies aggressive driver behaviours together with manoeuvre classification. By developing the PCNN method in the previous chapter, we has applied multi-task training by utilizing CNNs in parallel. However, the results illustrated that some spatial relations were missed by CNN due to locality. Rotating location data resolved the problem. Nevertheless, our purpose is to have fewer pre-processing stages and, more efficient and stable model. For this reason, we improved our PCNN model to the SPCNN-A model to capture more spatial relationships and to capture more global features with local features from time series data.

Chapter 6

Conclusion

6.1 Summary and Conclusion

In this research, we put forth a system for reducing features with the help of Sparse Auto Encoders (SAEs). This system has been designed to distinguish specific characteristics in human driving behaviours. Our system was tested using data obtained from smartphone sensors, demonstrating its ability to isolate key features that aid in identifying different driving maneuvers. This marks a crucial stride in the development of driving assist systems that could function in real-time, utilizing embedded sensors in smartphones and operating within a framework of limited computational resources. It is also privacy-preserving since the data we have utilized is non-invasive, not including face recording or other personal private data.

We further advanced our work by proposing a unique deep learning model we call the Parallel Convolutional Neural Network (PCNN). This model classifies aggressive driver behaviours as well as driving maneuvers. We conducted extensive testing of the PCNN using large datasets, both simulated and from real-world driving scenarios. The results demonstrate the PCNN's efficacy in identifying high-level features differentiating driving maneuvers and aggressive driver behaviours. The results also show that these CNN models can effectively work in parallel, mutually enhancing each other's performance.

Our final proposal in Chapter 5 is Spatial, Convolutional, Parallel Neural Network-A (SPCNN-A) a model that efficiently combines spatial, convolutional, self-attention and parallel components for simultaneous classification of aggressive driver behaviours and driving maneuvers. While PCNN did include multi-tasking by using parallel CNNs, it missed

capturing some spatial relations due to their locality. To tackle this issue, we developed the SPCNN-A model. This new model successfully captures more spatial relationships and combines local and global features extracted from time-series data. The SPCNN-A model provides a more robust solution with fewer pre-processing stages, leading to a more efficient and stable system, and taking us a step closer to designing advanced driving assist systems.

6.2 Thesis Contributions and Impact of the Study

The main contribution of this work is to evaluate the performance of multiple combinations of machine learning algorithms on time series data and simulated datasets to detect aggressive driving events. We perform a data simulation and a data-collecting phase from different sensors while performing different manoeuvres. We present how the CNN learning methods can be employed in the task and evaluate the accuracy of the technique aimed at classifying each behaviour. Although CNNs were originally proposed for static 2D images, they can be applied to time series efficiently by averaging features spatially for uniformly sized time windows. In these works, CNNs learn useful features directly from data. This study has shown that learned features from CNNs can perform efficiently by using position, velocity and angle features from large simulated datasets. We may infer that CNNs can also learn useful features from in-vehicle and smartphone sensor data and use these features for successful classification if the data is sufficiently large. In this study, we are trying to overcome the large-scale classification problem by using SPCNN-A. We have preferred to utilize CNNs rather than RNNs since time window lengths are short and we may not need long-term memory for training. In order to investigate whether the hypothesis behind our preference is true or not, PCNN has been compared with RNNs. The proposed approach can be used for ADAS systems to analyze and monitor driver behaviour. To the best of our knowledge, this is the first work of extending PCNN classification for driving behaviours directly using position, velocity and angle features. Experiments on a large dataset are expected to prove that, in terms of identifying driver behaviour, the proposed deep learning approach significantly saves time and increases accuracy. This indicates that SPCNN-A deep learning can be a powerful tool for characterizing aggressive driving styles.

This thesis makes several contributions to the area. These contributions are summarized below;

- The presented work provides a comprehensive analysis of aggressive driver behaviours. We have investigated manoeuvres, and aggressive and normal driver behaviours de-

rived from a driven dataset and a big dataset that is simulated by using SUMO and Webots respectively.

- The large simulated dataset is created by random selection from options, e.g. selecting random regions from the world, choosing a random distribution of type of vehicles, and driver behaviour parameters. This provides a big large-scale simulated dataset.

- We proposed to use a Parallel-CNN method for time series data analysis. This study is the first attempt to figure out aggressive driver behaviour by using shared layers of CNNs for multiple tasks.

- We improved the PCNN method by adding SpatialCNN and the attention layer. It is used for time series data analysis. This study is the first attempt to figure out aggressive driver behaviour by using shared layers of SCNNs with attention for multiple tasks.

- The proposed PCNN and SPCNN-A method works efficiently for driver behaviour classification tasks.

- Combined simulation procedure (SUMO + Webots) is automated and randomized. Therefore the simulated dataset is unique and large-scale.

- The large real-world driving DBL data is introduced for Driver behaviour learning.

- We have introduced the idea of the Spatial CNN with attention(SPCNN-A) model on time series in order to improve CNN's feature extraction capability on time series which include spatial information.

- We apply PCNN on a real driven dataset which is collected for the DBL project.

- We apply SPCNN-A on DBL data to classify aggressive and normal driver behaviours.

- This study aims to utilize data which can be collected cost-efficiently. Although we will be using in-vehicle sensors to validate the method, the method can be applicable to the data which is collected by using basic smartphone sensors.

- The possible application of this study can be in an ADAS for increased safety or a driver assessment system for determining insurance.

6.3 Future works

One possible direction could be to broaden the range of driving behaviours considered by the model. This could include not just aggressive patterns such as turns, lane changes, stops, merged and exiting highways but also more subtle and diverse maneuvers such as u-turns, accelerating and deceleration etc. Another possible approach could be to transition the current classification model into a regression model. This would allow for a real-time prediction of varying degrees of aggressive driving, providing a continuous, rather than discrete, output. A potentially beneficial direction might be the implementation of ensemble learning methods. This would involve combining various learning algorithms, such as CNN models with Recurrent Neural Networks (RNNs) or Long Short-Term Memory (LSTM) models, to enhance predictive performance. Instead of a binary classification of 'normal' or 'aggressive' driving, a future iteration of the model could discern between multiple categories or scales of behaviour, potentially including 'very safe', 'safe', 'normal', 'aggressive', and 'very aggressive'. Adding more data variables like wheel holding pressure and brake frequency could potentially improve the model's accuracy. At the same time, privacy could be maintained with this type of low-cost data. Given the time-intensive nature of manual labelling, exploring other methods, such as semi-supervised learning, active learning, and unsupervised learning, could be a useful future direction. These techniques would allow for more efficient labelling of larger datasets and enable the model to become more autonomous in its learning and labelling processes.

Acknowledgment

This project was possible through:

The financial support of grants from the National Science and Engineering Research Council (NSERC) of Canada (Grant #: `CRDPJ 519912 - 17`) and through financial support through our industry partner Magna International Inc. and,

The funding is provided by the Turkish Government.

References

[1] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang, "Spatial as deep: Spatial CNN for traffic scene understanding," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[2] S. Jetley, N. A. Lord, N. Lee, and P. H. S. Torr, "Learn To Pay Attention," *CoRR*, vol. abs/1804.02391, 2018.

[3] G. H. Walker and N. A. Stanton, *Human factors in automotive engineering and technology*. CRC Press, 2017.

[4] J. S. Hickman and E. S. Geller, "Self-management to increase safe driving among short-haul truck drivers," *Journal of Organizational Behavior Management*, vol. 23, no. 4, pp. 1–20, 2005.

[5] R. Fuller, "Towards a general theory of driver behaviour," *Accident Analysis & Prevention*, vol. 37, no. 3, pp. 461–472, 2005.

[6] J. Pérez, D. Gonzalez, and V. Milanés, "Vehicle Control in ADAS Applications: State of the Art," *Intelligent Transport Systems: Technologies and Applications*, p. 206, 2015.

[7] J. Paefgen, F. Kehr, Y. Zhai, and F. Michahelles, "Driving behavior analysis with smartphones: insights from a controlled field study," in *Proceedings of the 11th International Conference on mobile and ubiquitous multimedia*, p. 36, ACM, 2012.

[8] V. Di Lecce and M. Calabrese, "Nn-based measurements for driving pattern classification," in *Instrumentation and Measurement Technology Conference, 2009. I2MTC'09. IEEE*, pp. 259–264, IEEE, 2009.

[9] D. Johnson, M. M. Trivedi, *et al.*, "Driving style recognition using a smartphone as a sensor platform," in *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pp. 1609–1615, IEEE, 2011.

[10] D. Chen, K.-T. Cho, S. Han, Z. Jin, and K. G. Shin, "Invisible sensing of vehicle steering with smartphones," in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 1–13, ACM, 2015.

[11] P. Phondeenana, N. Noomwongs, S. Chatranuwathana, and R. Thitipatanapong, "Driving maneuver detection system based on GPS data," *Future Active Safety Technology Toward Zero Traffic Accidents, Nagoya*, 2013.

[12] S. Li, A. Oyedele, Z. Liu, H. He, and H. Yin, "Driver Behavior Recognition Using Vehicle Trajectory Data: A Survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 2, pp. 696–707, 2021.

[13] Y. Wang, D. Yang, H. Zhang, and Y. Zhang, "A survey on intelligent driving: from modeling driver behavior to optimal control for autonomous vehicles," *Frontiers of Information Technology & Electronic Engineering*, vol. 22, no. 2, pp. 213–233, 2021.

[14] Z. Chen, J. Yu, Y. Zhu, Y. Chen, and M. Li, "D 3: Abnormal driving behaviors detection and identification using smartphone sensors," in *2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pp. 524–532, IEEE, 2015.

[15] NHTSA, "Speeding ,Risky-driving," 2021.

[16] J. M. Houston, P. B. Harris, and M. Norman, "The Aggressive Driving Behavior Scale: Developing a self-report measure of unsafe driving practices," *North American Journal of Psychology*, vol. 5, no. 2, pp. 269–278, 2003.

[17] I. Mohamad, M. A. M. Ali, and M. Ismail, "Abnormal driving detection using real time global positioning system data," in *Space Science and Communication (IconSpace), 2011 IEEE International Conference on*, pp. 1–6, IEEE, 2011.

[18] A. Aljaafreh, N. Alshabatat, and M. S. N. Al-Din, "Driving style recognition using fuzzy logic," in *Vehicular Electronics and Safety (ICVES), 2012 IEEE International Conference on*, pp. 460–463, IEEE, 2012.

[19] D. Dörr, D. Grabengiesser, and F. Gauterin, "Online driving style recognition using fuzzy logic," in *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pp. 1021–1026, IEEE, 2014.

[20] J. F. Júnior, E. Carvalho, B. V. Ferreira, C. de Souza, Y. Suhara, A. Pentland, and G. Pessin, "Driver behavior profiling: An investigation with different smartphone sensors and machine learning," *PLoS one*, vol. 12, no. 4, p. e0174959, 2017.

[21] H. Zhao, H. Zhou, C. Chen, and J. Chen, "Join driving: A smart phone-based driving behavior evaluation system," in *Global Communications Conference (GLOBECOM), 2013 IEEE*, pp. 48–53, IEEE, 2013.

[22] J.-H. Hong, B. Margines, and A. K. Dey, "A smartphone-based sensing platform to model aggressive driving behaviors," in *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pp. 4047–4056, ACM, 2014.

[23] J. L. Deffenbacher, E. R. Oetting, and R. S. Lynch, "Development of a driving anger scale," *Psychological Reports*, vol. 74, no. 1, pp. 83–91, 1994.

[24] T. Lajunen and D. Parker, "Are aggressive people aggressive drivers? a study of the relationship between self-reported general aggressiveness, driver anger and aggressive driving," *Accident Analysis & Prevention*, vol. 33, no. 2, pp. 243–255, 2001.

[25] M. Conner, R. Lawton, D. Parker, K. Chorlton, A. S. Manstead, and S. Stradling, "Application of the theory of planned behaviour to the prediction of objectively assessed breaking of posted speed limits," *British Journal of Psychology*, vol. 98, no. 3, pp. 429–453, 2007.

[26] L. Aarts and I. Van Schagen, "Driving speed and the risk of road crashes: A review," *Accident Analysis & Prevention*, vol. 38, no. 2, pp. 215–224, 2006.

[27] D. Mitrovic, "Reliable method for driving events recognition," *IEEE Transactions on Intelligent Transportation systems*, vol. 6, no. 2, pp. 198–205, 2005.

[28] K. Karunanayake, K. D. Anjula, I. Perera, and H. Fernando, "Real-Time Driver Drowsiness Detection Using Deep Learning: A Review," *Applied Sciences*, vol. 11, no. 2, p. 502, 2021.

[29] G. A. M. Meiring and H. C. Myburgh, "A review of intelligent driving style analysis systems and related artificial intelligence algorithms," *Sensors*, vol. 15, no. 12, pp. 30653–30682, 2015.

[30] M. Fazeen, B. Gozick, R. Dantu, M. Bhukhiya, and M. C. González, "Safe driving using mobile phones," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 3, pp. 1462–1468, 2012.

[31] H. Zhao, L. Zhang, and H. Chen, "Driver behavior modeling and prediction for autonomous driving: A survey," *Information Fusion*, vol. 78, pp. 23–36, 2022.

[32] A. Kurt, J. L. Yester, Y. Mochizuki, and Ü. Özgüner, "Hybrid-state driver/vehicle modelling, estimation and prediction," in *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pp. 806–811, IEEE, 2010.

[33] A. Armand, D. Filliat, and J. Ibanez-Guzman, "Modelling stop intersection approaches using gaussian processes," in *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*, pp. 1650–1655, IEEE, 2013.

[34] L. M. Bergasa, D. Almería, J. Almazán, J. J. Yebes, and R. Arroyo, "Drivesafe: An app for alerting inattentive drivers and scoring driving behaviors," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pp. 240–245, IEEE, 2014.

[35] W. Chen, Y. Wang, D. Yang, and L. Zhu, "A survey of deep learning methods for driver behavior analysis," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[36] Y. Feng, Z. Yu, and X. Xu, "Deep Aggressive Driving Behavior Recognition Based on 3D CNN," in *2018 IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, pp. 500–504, IEEE, 2018.

[37] G. Castignani, T. Derrmann, R. Frank, and T. Engel, "Driver behavior profiling using smartphones: A low-cost platform for driver monitoring," *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 1, pp. 91–102, 2015.

[38] T. Toledo, O. Musicant, and T. Lotan, "In-vehicle data recorders for monitoring and feedback on drivers? behavior," *Transportation Research Part C: Emerging Technologies*, vol. 16, no. 3, pp. 320–331, 2008.

[39] H. Yang, F. Chen, Y. Liu, D. Yang, Y. Zhou, and Y. Du, "Smartphone-based driver behavior recognition using multi-sensor data fusion," *Applied Soft Computing*, vol. 89, p. 106108, 2020.

[40] G. Castignani, R. Frank, and T. Engel, "An evaluation study of driver profiling fuzzy algorithms using smartphones," in *2013 21st IEEE International Conference on Network Protocols (ICNP)*, pp. 1–6, IEEE, 2013.

[41] T. Chakravarty, A. Ghose, C. Bhaumik, and A. Chowdhury, "MobiDriveScore: a system for mobile sensor based driving analysis: A risk assessment model for improving one's driving," in *Sensing Technology (ICST), 2013 Seventh International Conference on*, pp. 338–344, IEEE, 2013.

[42] Y. Peng, G. Rong, and K. Li, "A survey on recent advances in vehicular network," *Digital Communications and Networks*, vol. 4, no. 3, pp. 163–176, 2018.

[43] J. Almazán, L. M. Bergasa, J. J. Yebes, R. Barea, and R. Arroyo, "Full auto-calibration of a smartphone on board a vehicle using IMU and GPS embedded sensors," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pp. 1374–1380, IEEE, 2013.

[44] V. Gadepally, A. Kurt, A. Krishnamurthy, and Ü. Özgüner, "Driver/vehicle state estimation and detection," in *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pp. 582–587, IEEE, 2011.

[45] N. Nikitas and M. M. Polycarpou, "A review on artificial intelligence methods for cooperative driving of mixed-autonomy vehicles," *Annual Reviews in Control*, vol. 53, pp. 417–439, 2022.

[46] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions," in *Robotics: Science and Systems*, vol. 2, 2016.

[47] X. Yang, Y. Chen, J. Wang, H. Liu, and D. Cao, "A cooperative driving strategy for mixed traffic of connected and automated vehicles and human-driven vehicles," *Transportation Research Part C: Emerging Technologies*, vol. 135, p. 103305, 2022.

[48] S. Singh, "Critical reasons for crashes investigated in the national motor vehicle crash causation survey," tech. rep., 2015.

[49] W. Deng, L. Zhao, S. Liu, Z. Li, and X. Chen, "Deep learning for vehicle behavior prediction," *Transportation Research Part C: Emerging Technologies*, vol. 104, pp. 321–334, 2019.

[50] T. Canada, "Canadian motor vehicle traffic collision statistics: 2012," 2014.

[51] A. Evans and S. Newey, "Rising to the Challenge: Addressing the Concerns of People on the Move," *Transport Reviews*, vol. 41, no. 1, pp. 1–19, 2021.

[52] T. Canada, "Road Safety in Canada," 2011.

[53] S. Jain, H. Sharma, L. Subramanian, A. Sreedevi, and A. Bhatnagar, "Aggressive driving: A survey of attitudes, opinions and behaviors," *Journal of Road Safety*, vol. 29, no. 1, pp. 3–20, 2018.

[54] M. Ablassmeier, K. Alvermann, R. Bierl, T. Doering, H. Foerg, M. Kemper, F. Koehne, C. Kress, M. Lenzhofer, M. Ritz, *et al.*, "Automotive smart display with integrated proximity and touch sensing," in *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 6432–6435, IEEE, 2007.

[55] A. Ng, J. Ngiam, C. Y. Foo, Y. Mai, and C. Suen, "Unsupervised feature learning and deep learning," 2013.

[56] Q. Wang, Z. Teng, B. Mehler, B. Reimer, and X. Fu, "Detecting driving events using smartphone sensors: A machine learning approach," *Transportation Research Part C: Emerging Technologies*, vol. 93, pp. 420–436, 2018.

[57] N. Lu, N. Cheng, N. Zhang, X. Shen, and J. W. Mark, "Connected vehicles: Solutions and challenges," *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 289–299, 2014.

[58] Z. Camlica, A. Hilal, and D. Kulić, "Feature abstraction for driver behaviour detection with stacked sparse auto-encoders," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 003299–003304, IEEE, 2016.

[59] Z. Camlica, J. Quesenberry, D. Carballo, and M. Crowley, "Aggressive Driver Behavior Detection using Parallel Convolutional Neural Networks on Simulated and Real Driving Data," in *2022 9th International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, pp. 1–8, IEEE, 2022.

[60] Z.-J. Cheng, L.-W. Jeng, and K. Li, "Behavioral Classification of Drivers for Driving Efficiency Related ADAS Using Artificial Neural Network," in *2018 IEEE International Conference on Advanced Manufacturing (ICAM)*, pp. 173–176, IEEE, 2018.

[61] V. Gadepally, A. Krishnamurthy, and U. Ozguner, "A framework for estimating driver decisions near intersections," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 2, pp. 637–646, 2014.

[62] F. Tango and M. Botta, "Real-time detection system of driver distraction using machine learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 894–905, 2013.

[63] A. K. Choudhary and P. K. Ingole, "Smart phone based approach to monitor driving behavior and sharing of statistic," in *Communication Systems and Network Technologies (CSNT), 2014 Fourth International Conference on*, pp. 279–282, IEEE, 2014.

[64] K. Takeda, J. H. Hansen, P. Boyraz, L. Malta, C. Miyajima, and H. Abut, "International large-scale vehicle corpora for research on driver behavior on the road," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1609–1623, 2011.

[65] A. Sathyanarayana, P. Boyraz, and J. H. Hansen, "Driver behavior analysis and route recognition by hidden markov models," in *Vehicular Electronics and Safety, 2008. ICVES 2008. IEEE International Conference on*, pp. 276–281, IEEE, 2008.

[66] M. Van Ly, S. Martin, and M. M. Trivedi, "Driver classification and driving style recognition using inertial sensors," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pp. 1040–1045, IEEE, 2013.

[67] A. Sathyanarayana, S. O. Sadjadi, and J. H. Hansen, "Leveraging sensor information from portable devices towards automatic driving maneuver recognition," in *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, pp. 660–665, IEEE, 2012.

[68] K. Li, L. Jin, Y. Jiang, H. Xian, and L. Gao, "Effects of driver behavior style differences and individual differences on driver sleepiness detection," *Advances in Mechanical Engineering*, vol. 7, no. 4, p. 1687814015578354, 2015.

[69] R. Bhoraskar, N. Vankadhara, B. Raman, and P. Kulkarni, "Wolverine: Traffic and road condition estimation using smartphone sensors," in *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*, pp. 1–6, IEEE, 2012.

[70] I. G. Daza, N. Hernández, L. M. Bergasa, I. Parra, J. J. Yebes, M. Gavilán, R. Quintero, D. F. Llorca, and M. Sotelo, "Drowsiness monitoring based on driver and driving data fusion," in *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pp. 1199–1204, IEEE, 2011.

[71] A. Demcenko, M. Tamosiunaite, A. Vidugiriene, and A. Saudargiene, "Vehicle's steering signal predictions using neural networks," in *Intelligent Vehicles Symposium, 2008 IEEE*, pp. 1181–1186, IEEE, 2008.

[72] H. Yang, Y. Wang, C. Xu, and H. Zhou, "Modeling and Prediction of Driver Lane-Changing Behaviors Using Long Short-Term Memory Networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 1, pp. 278–288, 2021.

[73] X. Zhang, H. Xu, Z. Li, and L. Li, "A Hybrid Model Combining Convolutional Neural Networks and Long Short-Term Memory Networks for Driver Drowsiness Detection," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[74] Q. Yang, J. Jia, Z. Zhao, F. Yang, Y. Gao, and Q. Wu, "Driver Behavior Analysis Framework based on Hidden Semi-Markov Models," in *2020 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 1–6, IEEE, 2020.

[75] N. Kuge, T. Yamamura, O. Shimoyama, and A. Liu, "A driver behavior recognition method based on a driver model framework," tech. rep., SAE Technical Paper, 2000.

[76] Y. Zhao, H. Zhao, L. Li, H. Zhang, and Y. Wang, "A Fuzzy Logic-Based Approach for Driving Style Recognition using Multiple Features," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 4776–4787, 2021.

[77] L. Wang, C. Li, Q. Li, C. Wu, X. Li, and Z. Wang, "An Improved DTW-based Approach for Driving Style Recognition using Smartphone Sensor Data," in *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1–6, IEEE, 2022.

[78] F. Liu, X. Chen, X. Hu, M. Wang, and Y. Wang, "Driving Event Recognition Based on Gaussian Mixture Models and Hidden Markov Models," in *2021 4th International Conference on Advanced Robotics and Intelligent Control (ICARIC)*, pp. 256–261, IEEE, 2021.

[79] X. Guo, Y. Xu, R. Yang, Y. Guo, and Y. Han, "SVM-Based Driver Drowsiness Detection Using Multi-Modal Data Fusion," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 2467–2472, IEEE, 2020.

[80] T. Li, H. Zhang, and H. Cheng, "A Novel SVM-Based Framework for Driver Behavior Prediction Using Deep Learning Features," *IEEE Access*, vol. 9, pp. 24436–24446, 2021.

[81] K. Chen, J. Yin, Z. Han, and R. Zhou, "Graph Convolutional Networks for Driver Behavior Analysis," in *2020 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 1–6, IEEE, 2020.

[82] S. Khan, H. Rahmani, S. A. A. Shah, and M. Bennamoun, "A guide to convolutional neural networks for computer vision," *Synthesis Lectures on Computer Vision*, vol. 8, no. 1, pp. 1–207, 2018.

[83] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[84] Y. LeCun, Y. Bengio, *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[85] H. Liu, T. Taniguchi, T. Takano, Y. Tanaka, K. Takenaka, and T. Bando, "Visualization of driving behavior using deep sparse autoencoder," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pp. 1427–1434, IEEE, 2014.

[86] A. Ng, "Cs294a lecture notes: Sparse autoencoder," *Lecture Notes: Sparse Autoencoder, Standford, CA, USA: Stanford Univ. Press*, pp. 1–19, 2011.

[87] C. Woo and D. Kulić, "Manoeuvre Segmentation Using Smartphone Sensors," in *Intelligent Vehicles Symposium(IV)*, pp. 572–577, IEEE, 2016.

[88] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2002.

[89] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[90] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using SUMO," in *The 21st IEEE International Conference on Intelligent Transportation Systems*, IEEE, 2018.

[91] Keras, "Keras documentation," 2019.

[92] N. Ketkar *et al.*, *Deep Learning with Python*. Springer, 2017.

[93] J. A. C. Dias, P. Machado, and F. C. Pereira, "Simulating the Impact of Driversâ Personality on City Transit,"

[94] S. M. Hegazy and M. N. Moustafa, "Classifying aggressive drivers for better traffic signal control," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 702–707, IEEE, 2017.

[95] W. Wang, W. Yang, A. Chen, and Z. Pan, "A Scoring Method for Driving Safety Credit Using Trajectory Data," *arXiv preprint arXiv:1811.12223*, 2018.

[96] M. Jensen, J. Wagner, and K. Alexander, "Analysis of in-vehicle driver behaviour data for improved safety," *International journal of vehicle safety*, vol. 5, no. 3, pp. 197–212, 2011.

[97] X. Liu, H. Mei, H. Lu, H. Kuang, and X. Ma, "A vehicle steering recognition system based on low-cost smartphone sensors," *Sensors*, vol. 17, no. 3, p. 633, 2017.

[98] R. Chhabra, S. Verma, and C. Rama Krishna, "Detecting Aggressive Driving Behavior using Mobile Smartphone," in *Proceedings of 2nd International Conference on Communication, Computing and Networking* (C. R. Krishna, M. Dutta, and R. Kumar, eds.), (Singapore), pp. 513–521, Springer Singapore, 2019.

[99] F. G. Habtemichael and L. de Picado Santos, "Crash risk evaluation of aggressive driving on motorways: Microscopic traffic simulation approach," *Transportation research part F: traffic psychology and behaviour*, vol. 23, pp. 101–112, 2014.

[100] Z. Ma, D. Chang, J. Xie, Y. Ding, S. Wen, X. Li, Z. Si, and J. Guo, "Fine-grained vehicle classification with channel max pooling modified CNNs," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3224–3233, 2019.

[101] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[102] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[103] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[104] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[105] B. Yang, L. Wang, D. Wong, L. S. Chao, and Z. Tu, "Convolutional self-attention networks," *arXiv preprint arXiv:1904.03107*, 2019.

[106] V. Zambaldi, D. Raposo, A. Santoro, V. Bapst, Y. Li, I. Babuschkin, K. Tuyls, D. Reichert, T. Lillicrap, E. Lockhart, *et al.*, "Relational deep reinforcement learning," *arXiv preprint arXiv:1806.01830*, 2018.

[107] I. Bello, B. Zoph, A. Vaswani, J. Shlens, and Q. V. Le, "Attention augmented convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3286–3295, 2019.

[108] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

[109] J. Hu, L. Shen, S. Albanie, G. Sun, and A. Vedaldi, "Gather-excite: Exploiting feature context in convolutional neural networks," in *Advances in Neural Information Processing Systems*, pp. 9401–9411, 2018.

[110] H. Xu and K. Saenko, "Ask, attend and answer: Exploring question-guided spatial attention for visual question answering," in *European Conference on Computer Vision*, pp. 451–466, Springer, 2016.

[111] E. Romera, L. M. Bergasa, and R. Arroyo, "Need data for driver behaviour analysis? Presenting the public UAH-DriveSet," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 387–392, IEEE, 2016.

[112] K. Saleh, M. Hossny, and S. Nahavandi, "Driving behavior classification based on sensor data fusion using LSTM recurrent neural networks," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–6, IEEE, 2017.

[113] Z. Ouyang, J. Niu, and M. Guizani, "Improved vehicle steering pattern recognition by using selected sensor data," *IEEE Transactions on Mobile Computing*, vol. 17, no. 6, pp. 1383–1396, 2017.

[114] B.-F. Wu, Y.-H. Chen, and C.-H. Yeh, "Fuzzy logic based driving behavior monitoring using hidden markov models," in *2012 12th International conference on ITS telecommunications*, pp. 447–451, IEEE, 2012.

[115] H. A. Ibrahim, A. K. Aly, and B. H. Far, "A system for vehicle collision and rollover detection," in *2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1–6, IEEE, 2016.

[116] A. H. Ali, A. Atia, and M.-S. M. Mostafa, "Recognizing driving behavior and road anomaly using smartphone sensors," *International Journal of Ambient Computing and Intelligence (IJACI)*, vol. 8, no. 3, pp. 22–37, 2017.

[117] W. Cao, X. Lin, K. Zhang, Y. Dong, S. Huang, and L. Zhang, "Analysis and evaluation of driving behavior recognition based on a 3-axis accelerometer using a random forest approach," in *Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pp. 303–304, 2017.

[118] C. Saiprasert, T. Pholprasit, and W. Pattara-Atikom, "Detecting driving events using smartphone," in *Proceedings of the 20th ITS World Congress*, vol. 11, p. 11, 2013.

[119] H. Sharma, R. K. Reddy, and A. Karthik, "S-CarCrash: Real-time crash detection analysis and emergency alert using smartphone," in *2016 International Conference on Connected Vehicles and Expo (ICCVE)*, pp. 36–42, IEEE, 2016.

[120] L. Sadouk, "CNN Approaches for Time Series classification," in *Convolutional Neural Network*, IntechOpen, 2018.

[121] T. T. Um, V. Babakeshizadeh, and D. Kulić, "Exercise motion classification from large-scale wearable sensor data using convolutional neural networks," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2385–2390, IEEE, 2017.

[122] C. Ou, C. Ouali, S. M. Bedawi, and F. Karray, "Driver Behavior Monitoring Using Tools of Deep Learning and Fuzzy Inferencing," in *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 1–7, IEEE, 2018.

[123] Y. Tu, F. Zhang, and Z. Wang, "Based on Hidden Markov Model to Identify the Driver Lane-Changing Behavior of Automobile OBD Internet of Vehicles Research and Design," in *Advances in Computer and Computational Sciences*, pp. 257–263, Springer, 2018.

[124] J. Bärgman, J. Werneke, C.-N. Boda, J. Engström, K. Smith, and A. Volvo, "Using manual measurements on event recorder video and image processing algorithms to extract optical parameters and range," in *Seventh International Driving Symposium on Human Factors in Driver Assessment, Training, and Vehicle Design, Bolton Landing*, 2013.

[125] C.-Y. Chan, "On the detection of vehicular crashes-system characteristics and architecture," *Vehicular Technology, IEEE Transactions on*, vol. 51, no. 1, pp. 180–193, 2002.

[126] P. Needham, "Collision prevention: The role of an accident data recorder (ADR)," 2001.

[127] O. Yarok, "An In-Vehicle Data Recorder for Evaluation of Driving Behavior and Safety," 2006.

[128] O. Musicant, T. Lotan, and T. Toledo, "Safety Correlation and Implications of In-Vehicle Data Recorder on Driver Behavior," in *Transportation Research Board 86th Annual Meeting*, no. 07-2173, 2007.

[129] M. Ueyama, S. Ogawa, H. Chikasue, and K. Muramatu, "Relationship between driving behaviour and traffic accidents-accident data recorder and driving monitor recorder," in *Proceedings: International Technical Conference on the Enhanced Safety of Vehicles*, vol. 1998, pp. 402–409, National Highway Traffic Safety Administration, 1998.

[130] Y. Arai, T. Nishimoto, Y. Ezaka, and K. Yoshimoto, "Accidents and near-misses analysis by using video drive-recorders in a fleet test," in *Proceedings of the 17th ESV conference, Amsterdam*, pp. 01–S4, 2001.

[131] E. Green, K. Agent, and J. Pigman, "Evaluation of auto incident recording system," *Kentucky Transportation Cabinet*, 2005.

[132] I. Han and K. Yang, "Characteristic analysis for cognition of dangerous driving using automobile black boxes," *International journal of automotive technology*, vol. 10, no. 5, pp. 597–605, 2009.

[133] M. Lan, M. Rofouei, S. Soatto, and M. Sarrafzadeh, "SmartLDWS: A robust and scalable lane departure warning system for the smartphones," in *Intelligent Transportation Systems, 2009. ITSC'09. 12th International IEEE Conference on*, pp. 1–6, IEEE, 2009.

[134] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: rich monitoring of road and traffic conditions using mobile smartphones," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pp. 323–336, ACM, 2008.

[135] C. Thompson, J. White, B. Dougherty, A. Albright, and D. C. Schmidt, "Using smartphones to detect car accidents and provide situational awareness to emergency responders," in *Mobile Wireless Middleware, Operating Systems, and Applications*, pp. 29–42, Springer, 2010.

[136] S.-H. Chen, J.-S. Pan, and K. Lu, "Driving behavior analysis based on vehicle OBD information and AdaBoost algorithms," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. 1, 2015.

[137] M. Sundbom, P. Falcone, and J. Sjöberg, "Online driver behavior classification using probabilistic ARX models," in *16th International IEEE Conference on Intelligent Transportation Systems, October 2013, The Hague, The Netherlands*, pp. 1107–1112, 2013.

[138] G. Xu, L. Liu, and Z. Song, "Driver behavior analysis based on bayesian network and multiple classifiers," in *Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on*, vol. 3, pp. 663–668, IEEE, 2010.

[139] Y. Han, Y. Yao, and H. Liu, "Design and Implementation of Driving Behavior Analysis System," *Journal of Networks*, vol. 9, no. 8, pp. 2114–2120, 2014.

[140] A. Liu and D. Salvucci, "Modeling and prediction of human driver behavior," in *Intl. Conference on HCI*, 2001.

[141] P. Angkititrakul, C. Miyajima, and K. Takeda, "An improved driver-behavior model with combined individual and general driving characteristics," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pp. 426–431, IEEE, 2012.

[142] P. Angkititrakul, C. Miyajima, and K. Takeda, "Modeling and adaptation of stochastic driver-behavior model with application to car following," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pp. 814–819, IEEE, 2011.

[143] C. Miyajima, Y. Nishiwaki, K. Ozawa, T. Wakita, K. Itou, K. Takeda, and F. Itakura, "Driver modeling based on driving behavior and its evaluation in driver identification," *Proceedings of the IEEE*, vol. 95, no. 2, pp. 427–437, 2007.

[144] T. V. Mathew and K. Ravishankar, "Neural network based vehicle-following model for mixed traffic conditions," 2012.

[145] K. Ravishankar and T. V. Mathew, "Vehicle-type dependent car-following model for heterogeneous traffic conditions," *Journal of transportation engineering*, vol. 137, no. 11, pp. 775–781, 2011.

[146] A. Khodayari, A. Ghaffari, R. Kazemi, F. Alimardani, and R. Braunstingl, "Improved adaptive neuro fuzzy inference system car-following behaviour model based on the driver-vehicle delay," *Intelligent Transport Systems, IET*, vol. 8, no. 4, pp. 323–332, 2014.

[147] T. Sato and M. Akamatsu, *Understanding Driver Car-Following Behavior Using a Fuzzy Logic Car-Following Model*. INTECH Open Access Publisher, 2012.

[148] A. Kusuma, L. Ronghui, and F. Montgomery, "Gap Acceptance Behaviour in Motorway Weaving Sections," in *Proceedings of the Eastern Asia Society for Transportation Studies*, vol. 9, 2013.

[149] H. Farah, S. Bekhor, A. Polus, and T. Toledo, "A passing gap acceptance model for two-lane rural highways," *Transportmetrica*, vol. 5, no. 3, pp. 159–172, 2009.

[150] T. Toledo, H. Koutsopoulos, and M. Ben-Akiva, "Modeling integrated lane-changing behavior," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1857, pp. 30–38, 2003.

[151] T. Toledo, "Driving behaviour: models and challenges," *Transport Reviews*, vol. 27, no. 1, pp. 65–84, 2007.

[152] G. Xu, L. Liu, Y. Ou, and Z. Song, "Dynamic modeling of driver control strategy of lane-change behavior and trajectory planning for collision prediction," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 13, no. 3, pp. 1138–1155, 2012.

[153] X. He, J. Hu, J. Lu, M.-Y. Wu, and B. Guerin, "Driver lane changing behavior," in *Computer Science and Network Technology (ICCSNT), 2011 International Conference on*, vol. 4, pp. 2385–2389, IEEE, 2011.

[154] A. Cabrera, S. Gowal, and A. Martinoli, "A new collision warning system for lead vehicles in rear-end collisions," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pp. 674–679, Ieee, 2012.

[155] M. R. Hafner, D. Cunningham, L. Caminiti, and D. Del Vecchio, "Cooperative collision avoidance at intersections: Algorithms and experiments," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 14, no. 3, pp. 1162–1175, 2013.

[156] E. U. Warriach, M. Aiello, and K. Tei, "A machine learning approach for identifying and classifying faults in wireless sensor network," in *Computational Science and Engineering (CSE), 2012 IEEE 15th International Conference on*, pp. 618–625, IEEE, 2012.

[157] L. He, C.-f. Zong, and C. Wang, "Driving intention recognition and behaviour prediction based on a double-layer hidden markov model," *Journal of Zhejiang University SCIENCE C*, vol. 13, no. 3, pp. 208–217, 2012.

[158] A. Pentland and A. Liu, "Modeling and prediction of human behavior," *Neural computation*, vol. 11, no. 1, pp. 229–242, 1999.

[159] W. Takano, A. Matsushita, K. Iwao, and Y. Nakamura, "Recognition of human driving behaviors based on stochastic symbolization of time series signal," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 167–172, IEEE, 2008.

[160] P. Handel, I. Skog, J. Wahlstrom, F. Bonawiede, R. Welch, J. Ohlsson, and M. Ohlsson, "Insurance telematics: Opportunities and challenges with the smartphone solution," *Intelligent Transportation Systems Magazine, IEEE*, vol. 6, no. 4, pp. 57–70, 2014.

[161] H. Eren, S. Makinist, E. Akin, and A. Yilmaz, "Estimating driving behavior by a smartphone," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pp. 234–239, IEEE, 2012.

[162] S. Sekizawa, S. Inagaki, T. Suzuki, S. Hayakawa, N. Tsuchida, T. Tsuda, and H. Fujinami, "Modeling and recognition of driving behavior based on stochastic switched ARX model," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 8, no. 4, pp. 593–606, 2007.

[163] H. Ouda, "Okuda hybrid-system package," 2010.

[164] G. E. Hinton, T. J. Sejnowski, and D. H. Ackley, *Boltzmann machines: Constraint satisfaction networks that learn*. Carnegie-Mellon University, Department of Computer Science Pittsburgh, PA, 1984.

[165] P. Smolensky, "Information processing in dynamical systems: Foundations of harmony theory," tech. rep., DTIC Document, 1986.

[166] R. Schubert, K. Schulze, and G. Wanielik, "Situation assessment for automatic lane-change maneuvers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 3, pp. 607–616, 2010.

[167] J. E. Naranjo, C. Gonzalez, R. Garcia, and T. De Pedro, "Lane-change fuzzy control in autonomous vehicles for the overtaking maneuver," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 3, p. 438, 2008.

[168] X. Yan, E. Radwan, and D. Guo, "Effects of major-road vehicle speed and driver age and gender on left-turn gap acceptance," *Accident Analysis & Prevention*, vol. 39, no. 4, pp. 843–852, 2007.

[169] K. Fitzpatrick, W. Schneider, and H. William, "Turn speeds and crashes within right-turn lanes," tech. rep., Texas Transportation Institute, Texas A & M University System, 2005.

[170] T. Sato and M. Akamatsu, "Modeling and prediction of driver preparations for making a right turn based on vehicle velocity and traffic conditions while approaching an intersection," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 11, no. 4, pp. 242–258, 2008.

[171] A. Barth and U. Franke, "Tracking oncoming and turning vehicles at intersections," in *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pp. 861–868, IEEE, 2010.

[172] A. Houenou, P. Bonnifait, V. Cherfaoui, and W. Yao, "Vehicle trajectory prediction based on motion model and maneuver recognition," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 4363–4369, IEEE, 2013.

[173] A. Doshi and M. M. Trivedi, "Examining the impact of driving style on the predictability and responsiveness of the driver: Real-world and simulator analysis," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pp. 232–237, IEEE, 2010.

[174] J. C. McCall, D. P. Wipf, M. M. Trivedi, and B. D. Rao, "Lane change intent analysis using robust operators and sparse bayesian learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 3, pp. 431–440, 2007.

[175] A. Jain, H. S. Koppula, B. Raghavan, and A. Saxena, "Know before you do: Anticipating maneuvers via learning temporal driving models," tech. rep., Cornell Univ Ithaca NY Dept of Computer Science, 2015.

[176] G. Reymond, A. Kemeny, J. Droulez, and A. Berthoz, "Role of lateral acceleration in curve driving: Driver model and experiments on a real vehicle and a driving simulator," *Human factors*, vol. 43, no. 3, pp. 483–495, 2001.

[177] C. Hatipoglu, U. Ozguner, and K. A. Redmill, "Automated lane change controller design," *IEEE transactions on intelligent transportation systems*, vol. 4, no. 1, pp. 13–22, 2003.

[178] P. B. Harris, J. M. Houston, J. A. Vazquez, J. A. Smither, A. Harms, J. A. Dahlke, and D. A. Sachau, "The Prosocial and Aggressive Driving Inventory (PADI): A self-report measure of safe and unsafe driving behaviors," *Accident Analysis & Prevention*, vol. 72, pp. 1–8, 2014.

[179] J. Healey, R. W. Picard, *et al.*, "Detecting stress during real-world driving tasks using physiological sensors," *IEEE Transactions on intelligent transportation systems*, vol. 6, no. 2, pp. 156–166, 2005.

[180] S. Al-Sultan, A. H. Al-Bayatti, and H. Zedan, "Context-aware driver behavior detection system in intelligent transportation systems," *IEEE transactions on vehicular technology*, vol. 62, no. 9, pp. 4264–4275, 2013.

[181] R. Kalsoom and Z. Halim, "Clustering the driving features based on data streams," in *Multi Topic Conference (INMIC), 2013 16th International*, pp. 89–94, IEEE, 2013.

[182] T. Hülnhagen, I. Dengler, A. Tamke, T. Dang, and G. Breuel, "Maneuver recognition using probabilistic finite-state machines and fuzzy logic," in *Intelligent vehicles symposium (IV), 2010 IEEE*, pp. 65–70, IEEE, 2010.

[183] A. Kurt and Ü. Özgüner, "A probabilistic model of a set of driving decisions," in *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pp. 570–575, IEEE, 2011.

[184] P. Boyraz, M. Acar, and D. Kerr, "Signal modelling and hidden markov models for driving manoeuvre recognition and driver fault diagnosis in an urban road scenario," in *Intelligent Vehicles Symposium, 2007 IEEE*, pp. 987–992, IEEE, 2007.

[185] N. Oliver and A. P. Pentland, "Graphical models for driver behavior recognition in a smartcar," in *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*, pp. 7–12, IEEE, 2000.

[186] H. Okuda, N. Ikami, T. Suzuki, Y. Tazaki, and K. Takeda, "Modeling and analysis of driving behavior based on a probability-weighted ARX model," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 98–112, 2013.

[187] L. Chong, M. M. Abbas, A. M. Flintsch, and B. Higgs, "A rule-based neural network approach to model driver naturalistic behavior in traffic," *Transportation Research Part C: Emerging Technologies*, vol. 32, pp. 207–223, 2013.

[188] H. Fawzi and O. Fawzi, "Towards a Formal Verification of the Robustness of Neural Networks Against Adversarial Examples," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2937–2950, 2021.

[189] A. Borning, B. Friedman, J. Davis, and P. Lin, "Human-computer interaction and autonomous vehicles," in *The Oxford Handbook of Ethics of AI*, pp. 309–324, Oxford University Press, 2021.

[190] M. Mihajlov and V. Dodevska, "Fatigue Detection System Based on Eye Blink Duration Using OpenCV," in *ICT Innovations 2019*, pp. 241–248, Springer, Cham, 2020.

[191] D. P. Brumby and A. L. Cox, "Towards safer interactions at the interface: A workshop on driving and HCI," in *Extended Abstracts of the 2017 CHI Conference on Human Factors in Computing Systems*, pp. 280–283, ACM, 2017.

[192] S. Pan and D. Zhang, "Human-machine interaction in the era of Internet of Things: a review," in *2017 International Conference on Orange Technologies (ICOT)*, pp. 29–32, IEEE, 2017.

[193] A. Doshi and M. M. Trivedi, "On the roles of macroscopic and microscopic models in the study of driver behavior under cognitive loading," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 485–497, 2011.

[194] Y. Wang, Y. Hu, and G. Wen, "Emotion Recognition From Multi-channel Physiological Signals for Driver Monitoring System: A Review," *IEEE Access*, vol. 9, pp. 142188–142200, 2021.

[195] C.-C. Chen, S. Li, Y. Wang, and K. Zheng, "Vehicle-to-infrastructure-based real-time traffic information provision: A comprehensive survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 8, pp. 3137–3153, 2020.

[196] Z. Yang, K. Yang, X. Bai, H. Yu, and F. Xia, "Vehicle-to-pedestrian communication for vulnerable road user safety: A comprehensive survey," *Vehicular Communications*, vol. 23, p. 100214, 2020.

[197] P. Lin, "Why ethics matters for autonomous cars," in *Autonomous Driving*, pp. 69–85, Springer, Berlin, Heidelberg, 2016.

[198] M. Gerla, E.-K. Lee, G. Pau, and U. Lee, "Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds," in *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pp. 241–246, IEEE, 2014.

[199] A. Caliskan, J. J. Bryson, and A. Narayanan, "Semantics derived automatically from language corpora contain human-like biases," *Science*, vol. 356, no. 6334, pp. 183–186, 2017.

[200] J. K. Gurney, "Sue my car not me: Products liability and accidents involving autonomous vehicles," *JL Econ. & Pol'y*, vol. 9, p. 247, 2013.

[201] D. J. Fagnant and K. Kockelman, "Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations," *Transportation Research Part A: Policy and Practice*, vol. 77, pp. 167–181, 2015.

[202] D. Shinar, "Aggressive driving: the contribution of the drivers and the situation," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 1, no. 2, pp. 137–160, 1998.