Enhancing Cross-Dataset Performance in Distracted Driver Detection using Body Part Activity Recognition

Submitted by:
Frank Zandamela

Supervisor: Associate Professor Fred Nicolls
Co-Supervisor: Dr. Gene Stoltz
Department of Electrical Engineering
University of Cape Town

Submitted in fulfilment of the requirements for a *Master of Science in Electrical Engineering*

May 2024

Declaration of originality

I know the meaning of plagiarism and declare that all the work in the document, save for that which is properly acknowledged, is my own. This thesis/dissertation has been submitted to the Turnitin module (or equivalent similarity and originality checking software) and I confirm that my supervisor has seen my report and any concerns revealed by such have been resolved with my supervisor.

Signature: Signed by candidate .....
Name: Frank Zandamela
Date: 03 May 2024

Ethics Clearance Document

Application for Approval of Ethics in Research (EiR) Projects
Faculty of Engineering and the Built Environment, University of Cape Town

ETHICS APPLICATION FORM

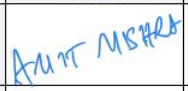
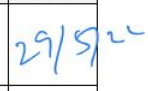
Please Note:
Any person planning to undertake research in the Faculty of Engineering and the Built Environment (EBE) at the University of Cape Town is required to complete this form **before** collecting or analysing data. The objective of submitting this application *prior* to embarking on research is to ensure that the highest ethical standards in research, conducted under the auspices of the EBE Faculty, are met. Please ensure that you have read, and understood the **EBE Ethics in Research Handbook** (available from the UCT EBE, Research Ethics website) prior to completing this application form: http://www.ebe.uct.ac.za/ebe/research/ethics1

APPLICANT'S DETAILS		
Name of principal researcher, student or external applicant		Frank Zandamela
Department		Department of Electrical Engineering
Preferred email address of applicant:		ZNDFRA003@myuct.ac.za
If Student	Your Degree: e.g., MSc, PhD, etc.	MSc (Eng) Electrical Engineering
	Credit Value of Research: e.g., 60/120/180/360 etc.	180
	Name of Supervisor (if supervised):	A/Professor Fred Nicolls
If this is a researchcontract, indicate the source of funding/sponsorship		
Project Title		Improving Distracted Driver Detection Algorithms Using Human Pose Estimation Features

I hereby undertake to carry out my research in such a way that:

- there is no apparent legal objection to the nature or the method of research; and
- the research will not compromise staff or students or the other responsibilities of the University;
- the stated objective will be achieved, and the findings will have a high degree of validity;
- limitations and alternative interpretations will be considered;
- the findings could be subject to peer review and publicly available; and
- I will comply with the conventions of copyright and avoid any practice that would constitute plagiarism.

APPLICATION BY	Full name	Signature	Date
Principal Researcher/ Student/External applicant	Frank Zandamela		09/03/2022
SUPPORTED BY	Full name	Signature	Date
Supervisor (where applicable)	Fred Nicolls		10/03/2022

APPROVED BY	Full name	Signature	Date
HOD (or delegated nominee) Final authority for all applicants who have answered NO to all questions in Section 1; and for all Undergraduate research (including Honours).			
Chair: Faculty EIR Committee For applicants other than undergraduate students who have answered YES to any of the questions in Section 1.			

Page 1 of 1

Abstract

Detecting distracted drivers is a crucial task, and the literature proposes various deep learning-based methods. Among these methods, convolutional neural networks dominate because they can extract and learn image features automatically. However, even though existing methods have reported remarkable results, the cross-dataset performance of these methods remains unknown. A problem arises because cross-dataset performance often indicates a model's generalisation ability. Without knowing the model's cross-dataset performance, deployment in the real world could result in catastrophic events.

This thesis investigates the generalisation ability of deep learning-based distracted driver detection methods. In addition, a robust distracted driver detection approach is proposed. The proposed approach is based on recognising distinctive activities of human body parts involved when a driver is operating a vehicle.

Representative state-of-the-art deep learning-based methods have been trained exclusively on three widely used image datasets and evaluated across the test sets of these datasets. Experimental results reveal that current deep learning-based methods for detecting distracted drivers do not generalise well on unknown datasets, particularly for convolutional neural network (CNN) models that use the entire image for prediction. In addition, the experiments indicated that although current distracted driver detection datasets are relatively large, they lack diversity.

The proposed approach was implemented using a state-of-the-art object detection algorithm called Yolov7. The cross-dataset performance of the implemented approach was evaluated on three benchmark datasets and a custom dataset. Experimental results demonstrate that the proposed approach improves cross-dataset performance. A cross-dataset accuracy improvement of 7.8% was observed. Most importantly, the overall balanced (F1-score) performance was improved by a factor of 2.68. The experimental results also revealed that although the proposed approach demonstrates commendable performance on a custom test set, all algorithms encountered challenges when dealing with the custom test set, mainly due to lower image quality and difficult lighting conditions.

The thesis presents two main contributions. Firstly, it evaluates the performance of current deep learning-based distracted driver detection algorithms across different datasets. Secondly, it proposes a robust algorithm for detecting distracted drivers by identifying key human body parts involved in operating a vehicle.

Acknowledgements

I want to express my deepest gratitude to all those who have contributed to the completion of this thesis. The journey was filled with challenges and opportunities, and I am thankful to have had the support of so many wonderful individuals.

First and foremost, I wish to thank my mother, N'wa Mhlanga, for her unconditional support and selflessness, which allowed me to dedicate time to complete this thesis while she was taking care of my son, Enelo.

I want to thank my supervisors, Professor Fred Nicolls and Dr. Gene Stoltz, for their invaluable input, guidance, and support to assist me in completing this thesis.

I thank the CSIR for the resources, time, and financial assistance to complete this thesis. I also wish to thank the CSIR's Optronic Sensor Systems Impact Area and Dumisani Kunene for allowing me to use their computational resources and the CSIR dataset used in this study.

I would like also to thank my girlfriend, N'wa Ntshana, for her support.

Finally, I would like to thank Dr. Michael Harmse and Dr. Rigardt Coetzee for proofreading this thesis.

Dedication

This thesis is dedicated to my mother, N'wa Mhlanga.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Road accidents causing fatalities are a major global issue, impacting society and the economy [1]. The impact is particularly significant in developing countries such as South Africa, where distracted driving is a leading cause [2]. It is generally agreed that if driver distractions could be measured and addressed quickly through alerts, crash rates could be reduced [3]–[5]. Consequently, effective driver distraction measurement methods are crucial to address the issue of distracted driving.

Various driver distraction measurement methods have been proposed in the literature. However, these methods have their limitations. In particular, while deep learning-based methods have shown promise, they struggle with cross-dataset performance. This thesis aims to evaluate and assist in improving deep learning-based detection of distracted drivers using human body part action recognition. The study aims to fill gaps in the current literature and enhance the effectiveness of detecting distracted drivers for safer roads.

This chapter provides background to the identified research problems and reviews existing methods and their shortcomings. The fundamental research questions are articulated, serving as the foundation for this study. Moreover, the research goal and objectives that address the identified research problems are provided, along with an overview of the selected methodology to solve the identified problems and the research study's objectives.

Furthermore, the chapter provides an overview of the research study's outcomes and contributions. Additionally, the research outputs of this study are summarized. Finally, the chapter provides the scope of the research study. The chapter concludes by outlining the structure of the subsequent sections.

1.1 Problem statement

1.1.1 Background to research problems

Fatalities due to road accidents cost countries about 3% of their annual gross domestic product (GDP) [1]. The World Health Organization (WHO) reports that road accidents result in approximately 1.3 million deaths yearly. More than 90% of these fatalities come from developing countries

like South Africa [1]. South Africa is one of the African countries with the highest road traffic accidents [1]. In 2020, the South African Road Traffic Management Corporation (RTMC) reported that the total road traffic fatalities reached 12,503. Amongst others, a significant percentage results from driver distractions such as talking to the phone while driving and texting [2]. According to the National Highway Traffic Safety Administration (NHTSA), most road accidents and near-accidents can be accredited to driver distraction and lack of attention [6].

Due to the detrimental effects of distracted driving on society and the global economy, there is a need for driver distraction measurement methods. A consensus is that if driver distractions could be measured and addressed quickly through alerts, crash rates could be reduced [3]–[5]. Researchers have proposed various methods to measure driver distractions. These methods can be broadly categorised into four groups [7]:

- **Physiological methods:** Physiological methods measure driver distraction using the driver's physiological characteristics. Most research focuses on five physiological areas [8]: cardiac activity, respiratory activity, eye activity, speech measures, and brain activity.
- **Vehicle-based methods:** Vehicle-based methods use data about the vehicle's operation gathered from internal and external sources recorded by onboard diagnostic systems [9]. The data from the vehicle includes driving speed, steering wheel position, acceleration, and lane departure, among others.
- **Vision-based methods:** Vision-based methods, or behavioural methods, focus on the driver's behaviour captured using camera systems. These methods typically use image processing and computer vision techniques to monitor the driver's status. Most research analyses driver activities such as facial expression characteristics, eye closure, body posture, head posture, and hands on the wheel.
- **Hybrid methods:** Hybrid methods combine at least two of the above methods to develop a robust system that benefits from the features of the combined methods.

Deep learning's success in real-world vision problems has led to significant attention towards using convolutional neural networks (CNNs) to detect distracted drivers. Researchers favour CNNs because they can automatically extract image features and perform classification [10]. Promising results have been reported in the literature, demonstrating high accuracy within individual datasets (i.e., intra-dataset accuracy). However, despite the remarkable success, the performance of current deep learning-based vision methods remains limited in cross-dataset testing scenarios. Cross-dataset performance is crucial since it indicates a learning model's robustness and generalisation ability. The generalisation ability of a model gives a good indication of the model's likelihood of failure when deployed in a real-world system. Deploying a model in the real world without knowing its cross-dataset performance might result in catastrophic events. For example, a driver was fatally injured in an accident because a Tesla car crashed into a roadside barrier and caught fire while on autopilot [11].

1.1.2 Identified gaps

Few to no comprehensive studies evaluate distracted driver detection algorithms across datasets. Research papers on detecting distracted drivers using deep learning techniques often include com-

parative performance evaluations. However, such evaluations could be more comprehensive and consider cross-dataset performance.

It was also found through preliminary experiments that current deep learning distracted driver detection algorithms do not generalise well on unknown datasets, particularly CNN-based methods that utilise the entire image for prediction. For example, a ResNet50 model fine-tuned on a distracted driver detection image dataset obtained an intra-dataset classification accuracy of 95.91% but an average classification accuracy of 72% on the other three datasets not used for training.

The identified problems are **(1)** a need for comprehensive studies investigating the generalising ability of deep learning distracted driver detection algorithms and **(2)** a lack of robust deep learning distracted driver detection algorithms with consistent performance across different distracted driver detection image datasets.

1.2 Overview of existing methods

A literature survey was conducted to identify existing methods to solve the problems identified in Section 1.1.2. The focus was narrowed to three key areas to identify research topics and review existing methods. These areas include:

- **Deep learning-based distracted driver detection:** This focus area looked at research conducted to tackle the issue of detecting distracted drivers using deep learning, focusing on convolutional neural networks and static images.
- **Robust distracted driver detection:** This study area narrowed down the scope to only research that aimed at improving the generalisation ability of distracted driver detection.
- **Cross-dataset performance evaluation:** This area focused on whether existing research evaluates model generalisation ability across new image datasets.

Various academic databases, including IEEE Xplore, ScienceDirect, Springer, Wiley, and Google Scholar, were used to gather the literature reviewed in this study. The search used key terms such as robust distracted driver detection, deep learning, convolutional neural networks, cross-dataset performance, and generalization.

Table 1.1 provides an overview of the existing methods and their consideration for the three identified focus areas. The green check marks in the table indicate that a method meets the corresponding focus area of comparison. In contrast, the red crosses indicate specific focus areas that a method does not fulfil. Chapter 3 will present a detailed literature review of the existing methods.

Table 1.1: Overview of existing methods and identified research gaps.

Approaches	Sources	CNN-based approach	Robust distracted driver detection	Cross-dataset performance tested
CNN feature classification	[12]–[14]	✓	✗	✗
Ensemble	[15], [16]	✓	✓	✗
Data augmentation	[17], [18]	✓	✓	✗
CNN features + HOG features	[5], [19], [20]	✓	✓	✗
CNNs + RNNs	[21], [22]	✓	✓	✗
3D CNNs	[23], [24]	✓	✓	✗
Pose estimation	[25], [26]	✓	✓	✗
Driver segmentation	[27], [28]	✓	✓	✗
Driver ROI detection	[9], [29]–[31]	✓	✓	✗

Based on the literature presented in Table 1.1, it was found that much effort has been put into developing robust methods for detecting distracted drivers. However, it is still unclear how effective these approaches are in generalising to other datasets. Existing research studies do not evaluate their methods on other distracted driver detection datasets. Additionally, current approaches only utilize object detection to identify the driver's region of interest (ROI) and distraction objects like mobile phones and drinking bottles. None of the existing approaches focus on identifying crucial driver body parts, classifying their state into activities, and then making a final prediction.

1.3 Research questions

Reviewing existing methods (Section 1.2) gives rise to two primary search questions this study seeks to answer. These research questions are:

- **RQ1:** To what extent can deep learning distracted driver detection algorithms generalise on unknown image datasets not used for training?
- **RQ2:** Can the performance of CNN-based distracted driver detection be improved across different datasets by detecting driver body parts and classifying their state into activities?

1.4 Research goal and objectives

To answer the research questions stated in Section 1.3, the main goal of this study is to develop a robust approach based on human body part recognition. The approach is expected to enhance the ability of CNN-based distracted driver detection to generalise well.

To achieve the goal of the study, the following research objectives (ROJs) have been identified:

- **ROJ1:** To investigate current techniques for detecting distracted drivers using single static images with convolutional neural networks.

- **ROJ2:** To investigate the cross-dataset performance of deep learning distracted driver detection algorithms.
- **ROJ3:** To develop an approach that uses human part action recognition to improve distracted driver detection on unknown datasets.
- **ROJ4:** To evaluate the proposed approach using benchmark datasets and a custom dataset.

1.5 Overview of the methodology

This section provides an overview of the overall research methodology that is used to achieve the research objectives outlined in Section 1.4:

- **Deep learning in distracted driver detection (METHOD1):** As discussed in Section 1.2, to conduct the literature survey, the focus is first narrowed to three key areas: deep learning-based distracted driver detection, robust distracted driver detection, and cross-dataset performance evaluation. The three focus areas form the basis for the identification of research topics as well as reviewing existing methods. The literature reviewed is obtained from academic databases such as IEEE Xplore, ScienceDirect, and Google Scholar. The reviewed methods are then grouped and synthesised based on themes discovered in the literature, directly addressing ROJ1.
- **Cross-dataset performance evaluation (METHOD2):** The generalisation ability of a model refers to the variance in its performance when trained and fine-tuned on one or multiple datasets and subsequently tested on an entirely novel dataset, which the model has not previously encountered [32]. Consistent with this definition, METHOD2 will address ROJ2 by evaluating the cross-dataset performance of distracted driver detection algorithms using a three-step quantitative approach. Step 1 involves collecting image datasets and splitting the datasets into training, validation, and test sets. Step 2 involves selecting the algorithms and training them on each training set prepared in Step 1. The final step involves evaluating each algorithm on each of the test sets.
- **Robust distracted driver detection using human part action recognition (METHOD3):** Current image datasets are relatively large but not diverse. These datasets mainly come from experiments in simulators or real car environments [17], which produce images with similar backgrounds and within a narrow range of distracted driving scenarios. Creating a large and diverse image dataset is the apparent option from these observations. However, increasing the amount of data presents two challenges. First, considerable data is required to achieve notable performance improvement [33]. Second, acquiring large and diverse in-car driver poses for distracted driver detection is a challenge due to the reluctance of drivers to compromise their privacy and the cost associated with using multiple devices for data collection, not to mention the labour-intensive annotation process [34]. The challenges of increasing the training dataset also make the apparent solution of fine-tuning a model to a task-specific dataset less attractive [35]. As a result, METHOD3 will address ROJ3 by focusing on the key human body parts involved when a driver is operating a vehicle and using their activities to predict if a driver is distracted. The proposed approach is based on the hypothesis that focusing on only the key body parts of the driver will reduce input variance from dataset to dataset. Figure 1.1 provides an overview of the proposed approach.

- **Evaluation on benchmark and custom datasets (METHOD4):** To achieve ROJ4, the same cross-dataset performance evaluation approach will be utilised as the one outlined above for achieving ROJ2.

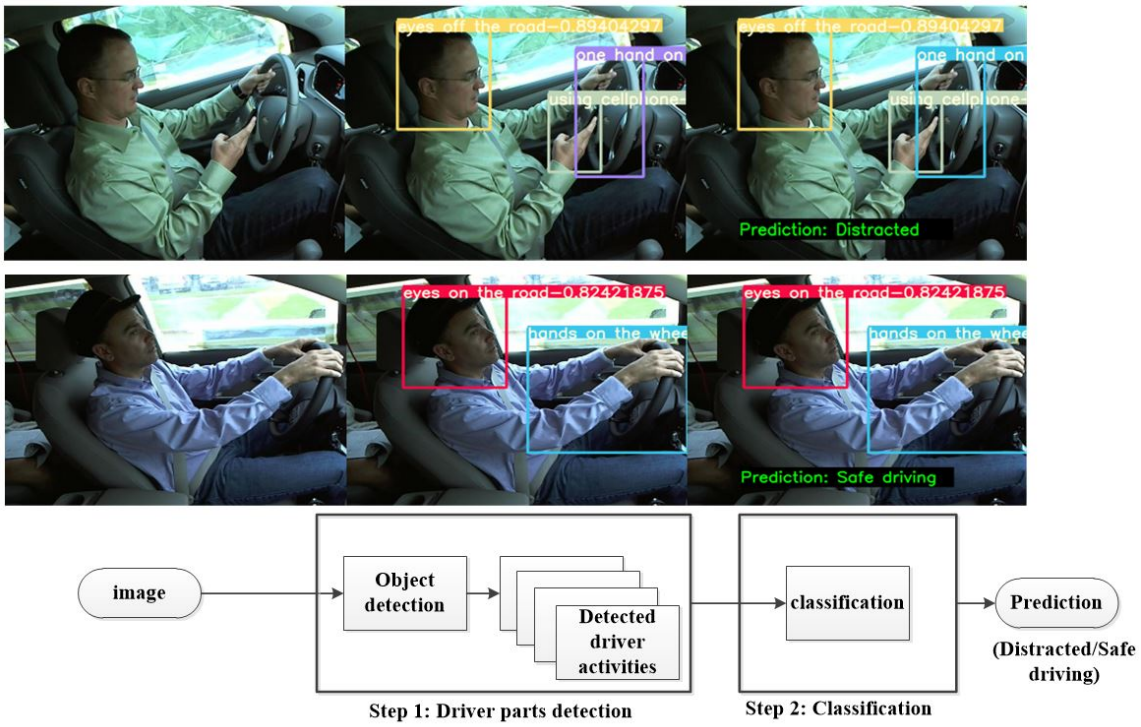


Figure 1.1: Proposed framework for distracted driver detection. **Step 1:** The driver's body parts are detected, and the activity of each body part is classified. **Step 2:** The detected activities are used to make a final prediction using a decision tree-based approach.

1.6 Overview of the outcomes

Using the methodology (METHOD1 to METHOD4) discussed in Section 1.5, the outcomes of this thesis can be summarised as follows:

- **ROJ1 outcomes:** Two major themes emerged from the literature survey. The themes include CNN feature classification and robust distracted driver detection. CNN feature classification methods extract driver features using a CNN and then use fully connected layers or a multilayer perceptron (MLP) to classify the driver behaviour. However, CNN feature extraction methods often overfit the training data and subsequently fail to generalise well on new data not used for training. To overcome the issue of overfitting, robust distracted driver detection methods use various approaches such as the ensemble approach, data augmentation, combining CNN features with HOG features, hybrid CNN-RNN approaches, 3-dimensional CNNs,

combining CNN features with human key points obtained using pose estimation algorithms, driver segmentation, and driver ROI detection.

- **ROJ2 outcomes:** Based on the cross-dataset performance evaluation results, it was found that, in general, deep learning distracted driver detection algorithms do not perform very well on testing datasets that do not come from the same dataset as the training dataset. This observation holds particularly true for CNN models that employ the entire image without removing background noise or utilising less variable features. In addition, it was found that although current distracted driver detection image datasets are large, they lack diversity – a characteristic that negatively affects their ability to generalise on data. As a result, models often resort to shortcut learning when trained on the current datasets.

- **ROJ3 outcomes:** The proposed approach (Figure 1.1) was successfully implemented using the Yolov7 [36] algorithm in three steps. In the first step, images randomly selected from a dataset were split into training, validation, and test sets. Images from the training and validation sets were manually annotated to highlight key human body parts involved when a driver operates a vehicle and common distraction objects such as cell phones and drinking bottles. In the second step, the Yolov7 model was adapted for the task of distracted driver detection using the source code provided by the Yolov7 authors on their official GitHub repository[1]. A standard Yolov7 model, pre-trained on the Microsoft COCO dataset [37], was fine-tuned to the annotated training dataset.

- **ROJ4 outcomes:** A classification accuracy improvement was observed, along with a significant overall balanced (F1-score) performance improvement. Further details regarding these improvements will be discussed in Chapter 6.

1.7 Contributions

The contributions of this thesis are derived from the review of existing methods presented in Section 1.2. Based on this review, it was found that existing approaches do not evaluate the generalisation ability of proposed methods, and no study evaluates the cross-dataset performance of existing distracted driver detection methods. In addition, existing driver ROI detection methods only use object detection to detect the driver ROI and common distraction objects such as cell phones and drinking objects. As a result, the methods proposed in this thesis (Section 1.5) to reach the study's research objectives (Section 1.4) will be new. The main contributions of the study are:

- **Evaluating the cross-dataset performance of current deep learning-based distracted driver detection algorithms:** Insights drawn from this study will indicate the readiness of existing methods for deployment in the real world. The study could also serve as a general guideline for selecting distracted driver detection algorithms and provide direction for future research.

- **Developing a robust distracted driver detection algorithm that relies on detecting key human body parts involved when a driver is operating a vehicle:** Even though existing research uses object detection to detect the driver ROI and common distractions [9], [29]–[31], a more robust approach that zooms into the key human body parts involved when

[1] https://github.com/WongKinYiu/yolov7

a driver is operating a vehicle is required to reduce the cross-dataset variance and thereby improve the generalisation ability of distracted driver detection.

- **Evaluating the cross-dataset performance of the proposed approach on benchmark datasets and a custom test set:** The cross-dataset performance of the proposed approach will indicate its generalisation ability, an important indicator for the readiness of a model for deployment and integration in real-world applications.

1.8 Research outputs

The contributions of this work have led to one published conference paper titled "*Cross-dataset performance evaluation of deep learning distracted driver detection algorithms*" [38] and a published journal article titled "*Enhancing Distracted Driver Detection with Human Body Activity Recognition using Deep Learning*" [39].

The first paper, published in the Proceedings of the 2022 RAPDASA-RobMech-PRASA-CoSAAMI conference, was the first work on cross-dataset performance evaluation on deep learning distracted driver detection algorithms. The paper's focus was evaluating and analysing the cross-dataset performance of state-of-the-art deep learning-based distracted driver detection approaches. This work demonstrated that, in general, the deep learning-based distracted driver detection algorithms do not generalise well on new data. It was also found that current distracted driver detection datasets are relatively large but not diverse. As a result, algorithms trained on these datasets often resort to shortcut learning, significantly reducing their ability to generalise to new data.

The journal article focuses on the development of a robust distracted driver detection approach by detecting and classifying the state of key human body parts that are involved when a driver is operating a vehicle. The paper forms part of Chapters 5 and 6.

1.9 Delimitations

This work must contend with the following delimitations:

- **Algorithms:** This study only focuses on vision-based deep learning distracted driver detection algorithms. In the research area of distracted driver detection, vision-based deep learning algorithms have demonstrated superior performance to traditional algorithms [19], [40]. In addition, due to financial constraints, algorithms using vehicle information will not be considered – only vision-based algorithms will be considered to avoid collecting and analysing vehicle data.
- **Input data:** This study only focuses on single static images instead of video data.
- **Performance:** The primary goal of this work is to improve the generalising ability of distracted driver detection. Therefore, the focus is only on improving distracted driver detection accuracy across different datasets. Other performance indicators, such as processing speed, are not considered.
- **Deployment:** Deployment and testing of the proposed approach in a real-world environment (car) are beyond the scope of this work.

1.10 Document structure

The structure of this document is as follows:

- **Chapter 2** provides background on distracted driving, fundamental concepts of convolutional neural networks, and brief details on the training of deep learning neural networks. In addition, the chapter covers the concepts of machine learning generalisation and regularisation. The chapter concludes by introducing object detection and fundamental object detection concepts relevant to this thesis.
- **Chapter 3** provides a detailed literature review on the key focus areas identified in Section 1.2. In addition, the chapter will introduce common datasets used to train distracted driver detection methods. The literature review will support the first research objective (ROJ1) outlined in Section 1.4.
- **Chapter 4** sets out to achieve the second objective (ROJ2) of this thesis. It evaluates how well deep learning-based distracted driver detection algorithms perform on unfamiliar image datasets not part of their training.
- **Chapter 5** focuses on achieving the third research objective (ROJ3) of this thesis. Specifically, the chapter focuses on developing a robust distracted driver detection approach based on recognising distinctive activities of human body parts involved when a driver is operating a vehicle.
- **Chapter 6** presents two experiments that focus on evaluating the robustness of the proposed approach. The activities carried out in this chapter support the fourth research objective (ROJ4) of this thesis.
- **Chapter 7** provides a conclusion to this study. The chapter concludes with a summary of the outcomes of this thesis and recommendations for future work.

Chapter 2

Background

This chapter sets the foundation for subsequent chapters by providing essential background knowledge. It briefly introduces distracted driver detection (Section 2.1) and the different methods used to measure distracted driving (Section 2.1.2). The chapter then introduces artificial neural networks (Section 2.2) and the fundamental building blocks of convolutional neural networks that form the basis for the methods introduced throughout this thesis (Section 2.3).

Subsequently, the chapter elaborates on the typical procedure for training deep neural networks (Section 2.4). Moreover, it delves into the concepts of machine learning generalisation and regularisation in Section 2.5. Section 2.6 introduces object detection and outlines fundamental concepts pertinent to this thesis. In Section 2.7, the details of the YOLOv7 architecture are presented. Finally, Section 2.8 concludes the chapter by furnishing a high-level overview of the implementation process for customising a YOLOv7 model, while also introducing commonly used performance metrics for evaluating object detectors.

2.1 Distracted driver detection

The purpose of this section is to provide a comprehensive overview of distracted driving detection methodologies, starting with a delineation of distracted driving, which categorises the causes of distraction. Following that, three fundamental methods of measuring distracted driving are discussed: physiological, vehicle-based, and vision-based. For each methodology, the technologies used are described, along with their advantages and disadvantages. The section concludes by focusing attention on vision-based approaches, primarily emphasising the use of deep learning methodologies utilising Convolutional Neural Networks (CNNs). Their commendable performance in real-world applications and promising outcomes in the field of distracted driver detection support this focus.

2.1.1 What is distracted driver detection

In the literature, the terms driver distraction and driver inattention are often used interchangeably. Although the two terms may refer to the same concept, this work will use the term "driver distraction". The National Highway Traffic Safety Administration (NHTSA) defines distracted driving as

any activity that draws away a driver's attention from the task of driving [6]. Distracted driving includes, amongst other non-driving activities, talking or texting on the phone, talking to passengers in the car, and eating and drinking. Engaging in non-driving activities is a potential distraction and constitutes unsafe driving. Driver distractions can be categorised into three groups as follows:

- **Visual distractions:** tasks that require the driver to take their eyes off the roadway.
- **Manual distractions:** tasks that require the driver to take their hands off the wheel.
- **Cognitive distractions:** mental workload associated with tasks or activities that take the driver's attention off driving.

2.1.2 Measurement methods for distracted driving

Chapter 1 briefly introduced three primary categories of methods for detecting distracted drivers: physiological, vehicle, and vision-based methods. The technology and techniques used in each of these categories are summarised as follows [27]:

- **Physiological-based methods:** Driver distraction can be detected by monitoring the driver's physical state using physiological-based methods. These methods assume that increased mental load leads to an increased physical response from the body [8]. Researchers typically focus on five physiological areas: cardiac activity, respiratory activity, eye activity, speech measures, and brain activity. Electro-physical devices such as electroencephalograph (EEG) [41], electrooculogram (EOG), and electromyograph (EMG) [42] are commonly used to measure the driver's physical state. While these methods have been reported to be highly accurate, some researchers argue that they can reduce user experience and increase hardware costs due to the need for wearable sensors [43].
- **Vehicle-based methods:** Vehicle-based methods rely on sensors to gather data about a vehicle's acceleration, deceleration, steering angle, heading angle, and speed. This data is then sent to a classifier model, such as a support vector machine (SVM) or multilayer perceptron (MLP) [44], which determines the driver's action based on the input data.
- **Vision-based methods:** Vision-based methods utilise vision sensors such as cameras to capture visual information about the driver through video or images. Most vision-based methods follow a two-step process, i.e., feature extraction and classification. Initially, researchers used hand-engineered low-level features such as the scale invariant feature transform (SIFT), speeded-up robust features (SURF), or histogram of oriented Gradients (HOG). These features are then passed to a classical machine learning model for classification. Commonly used classical machine learning models include support vector machines (SVM) [45] and random forest algorithms. However, in recent years, researchers have focused on using deep learning to solve the problem of distracted driver detection due to its success in other computer vision tasks such as number plate recognition [46]. Most deep learning-based distracted driver detection methods use convolutional neural networks (CNNs) due to their ability to learn features from images automatically.

Other methods that combine two of the three methods above to create hybrid methods have also been proposed in the literature. For example, Omerustaoglu *et al.* [47] proposed a hybrid method

combining a vehicle-based method with a vision-based one.

This thesis mainly explores vision-based techniques, particularly deep learning-based methods that utilize convolutional neural networks (CNNs). The reason for selecting CNNs is their proven track record in solving real-world vision problems like skin cancer classification [48] and fire and smoke detection [49]. Moreover, compared to traditional algorithms, CNNs have shown tremendous results in identifying distracted drivers [19], [40].

2.2 Artificial neural networks

Before introducing CNNs it is important to briefly introduce artificial neural networks (ANNs) since CNNs are simply ANNs that use convolution instead of general matrix multiplication in at least one of their layers [50].

ANNs are the foundation for developing deep learning models. ANNs are computer systems that mimic the connections between neurons in the human brain. The main objective of ANNs is to estimate a function that associates inputs with outputs using patterns derived from previously observed data.

A well-known and widely used type of artificial neural network is the multilayer perceptron (MLP) [51]. Figure 2.1 shows a typical architecture of an MLP. It is important to notice that every output neuron is connected to every input neuron. The MLP processes data in three steps. The input values are multiplied by individual weights (w_i) in the first step. In the second step, all weighted inputs are summed, and a bias term (b) is added to obtain (z), which is then passed through an activation function. The output of the neuron is the activated weighted sum and if a linear activation function is used, then the neuron's output ends up being the weighted sum z. Eq. 2.1 shows how the neuron's output is calculated for the single neuron shown in Figure 2.1:

$$z = b + \sum_i w_i \cdot x_i. \tag{2.1}$$

The final step is to pass the weighted sum, denoted as z, through an activation function. Activation functions serve a crucial purpose: they introduce non-linearity into the neural network [52]. The non-linearity introduced by activation functions is essential for addressing real-world problems, which frequently exhibit non-linear patterns. Moreover, activation functions enable artificial neural network (ANN) models to effectively represent and approximate complex functions. Additionally, activation functions play a role in regulating the flow of information within the network and improving its stability and convergence during the training process [53].

Figure 2.2 displays common activation functions used in deep neural networks. For an in-depth analysis of activation functions in deep neural networks, refer to [52].

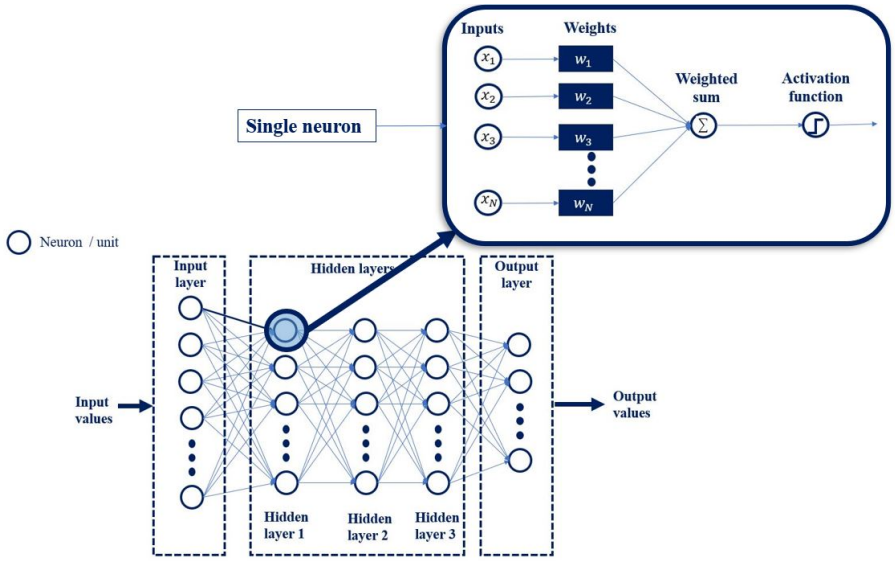


Figure 2.1: Typical structure of a multilayer perceptron. For a single neuron, input values are multiplied by the individual weights. The weighted sum is then computed and passed to the activation.

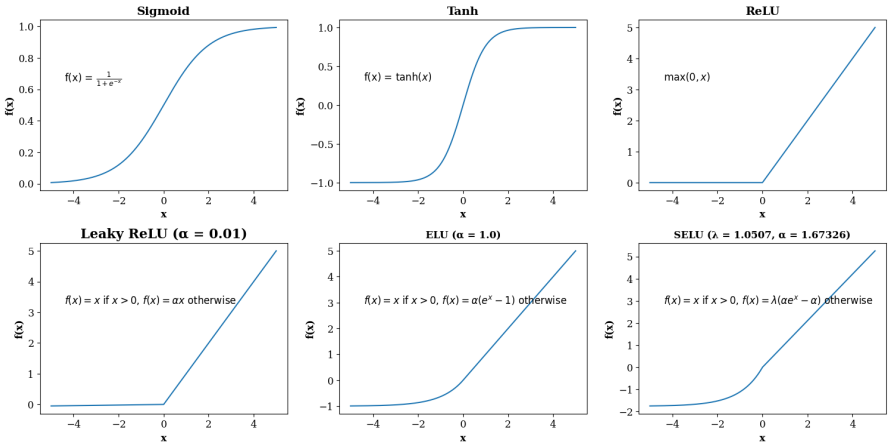


Figure 2.2: Commonly used activation functions in deep learning. **Top left:** Sigmoid activation function. **Mid-top:** Hyperbolic tangent. **Top right:** Rectified linear unit (commonly used activation function for deep neural networks). **Bottom left:** variant of the ReLU that allows for negative values. **Mid-bottom:** Exponential linear unit. **Bottom right:** Scaled exponential linear unit.

2.3 Convolutional neural networks

The success of AlexNet on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 marked a significant milestone and a breakthrough in deep learning by demonstrating the power of deep CNN architectures for image recognition tasks. However, the history of CNNs goes back to 1959 when David Hubel and Torsten Wiesel conducted groundbreaking experiments on the visual cortex of cats [54], identifying the presence of simple and complex cells that laid the foundation for understanding the hierarchical processing of visual information. In 1980, Fukushima then introduced a neural network model for pattern recognition called Neocognitron [55]. This paper introduced the concepts of feature extraction, pooling layers, and using convolutions in a neural network. Later, in 1998, Yann LeCun introduced LeNet-5 [56], a pioneering CNN architecture designed for handwritten digit recognition featuring convolutional layers and subsampling layers (pooling), which demonstrated the effectiveness of CNNs for image classification.

Convolutional neural networks are deep learning algorithms designed to specifically process and analyse data with a grid-like structure, such as images, videos, and audio spectrograms [50]. They have trainable architectures with multiple stages [57]. Each stage generates a set of arrays called feature maps, which serve as input and output. Typically, each stage consists of three layers: a convolutional layer, a non-linearity layer, and a feature pooling layer [57]. In addition to these three layers, fully connected (FC) layers, also known as dense layers, are commonly employed at the end of CNNs to use extracted features to make predictions or perform classification tasks. Additional layers, such as batch normalization (BN) and dropout layers, have also been introduced to improve CNN performance. The details of each layer are provided below.

2.3.1 Convolutional layer

A convolutional layer is a fundamental building block in a CNN. It applies a set of K learnable filters to the input volume and extracts low-level features through convolution. In deep learning, the convolution operation refers to an element-wise multiplication between two matrices followed by a sum [57]. In line with this definition, for CNNs convolution S is an element-wise multiplication between an input volume I and a two-dimensional (2D) filter K [50], [57],

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n K(i+m, j+n) I(m,n), \tag{2.2}$$

where m and n represent the dimensions of the filter K, and i, j represents the coordinates of a pixel in the input volume.

Each filter $(k_i j)$ applied in a convolutional layer has a size $M_k \times N_k$, where M_k is the height and N_k is the width and are nearly always square. The size of the filters, in terms of their spatial dimensions, is typically small but extends throughout the full depth of the input volume. The input volume has a width (n_3), height (n_2), and depth (n_1). Each filter slides across the input region, computes an element-wise multiplication, performs a summation, and then stores the output value in a 2-dimensional feature map with width and height given by m_3 and m_2, respectively. The depth of the input volume is equivalent to the number of channels in the image if the input is for the input layer of the CNN. For input volumes deeper in the CNN, the depth is equivalent to the number

of filters applied in the previous layer. Figure 2.3 illustrates the convolution operation in a typical convolutional layer.

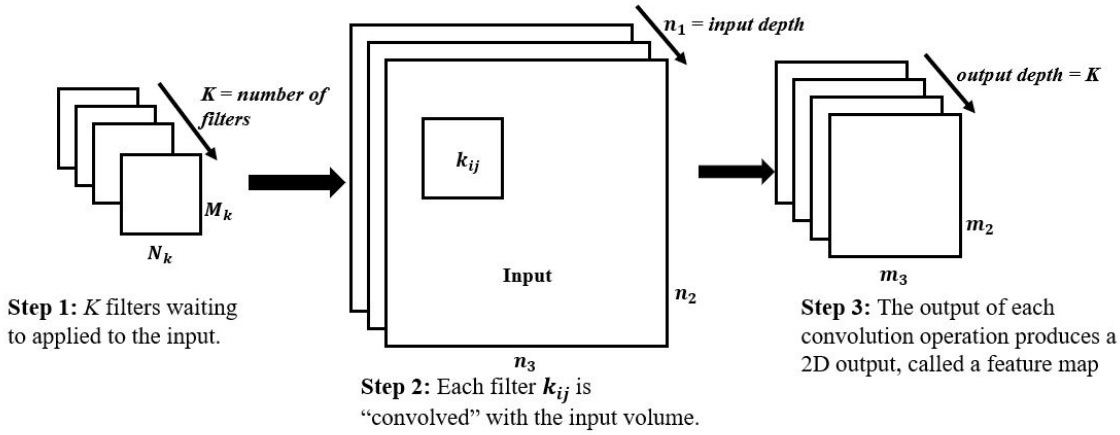


Figure 2.3: **Step 1:** At each convolutional layer in a CNN, K filters are applied to the input. **Step 2:** Each filter is convolved with the input volume. **Step 3:** The output of each convolution operation is a 2D feature map.

The convolution operation gives rise to three significant benefits of the convolutional layer [50]: *sparse connections*, *parameter sharing*, and *equivariant representations*. Due to sparse connections, neurons in the first convolutional layer do not connect to every pixel in the input image (like in the multilayer perceptron discussed in Section 2.2) but connect to pixels in their receptive fields. In turn, neurons in subsequent convolutional layers only connect to pixels in a small rectangle within the previous convolutional layer. Figure 2.4 illustrates the concept of sparse connections.

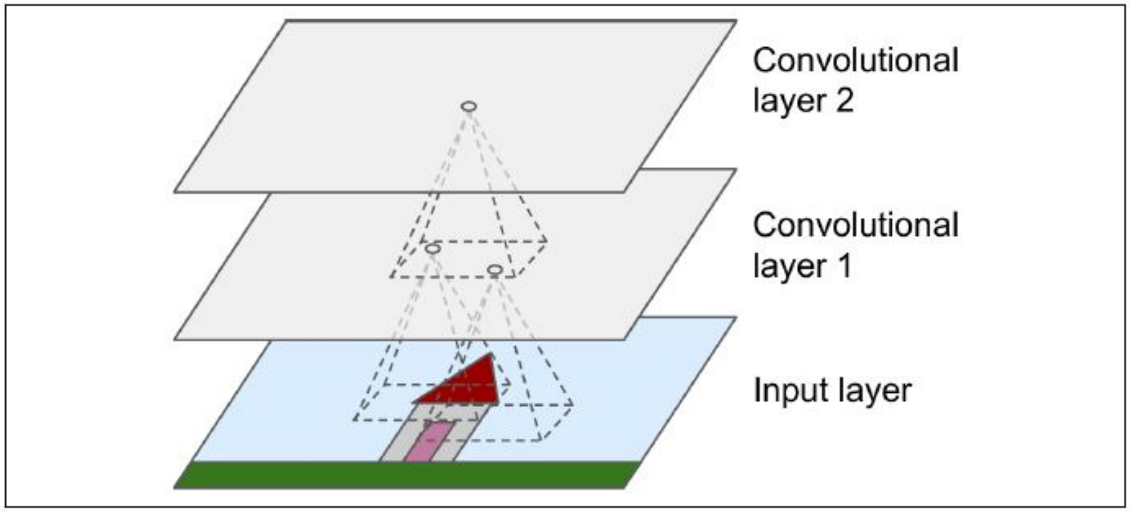


Figure 2.4: The concept of sparse connections and receptive field in a CNNs [58].

The concept of parameter sharing makes CNNs computationally and statistically efficient due to the smaller number of parameters that the network must learn. The sharing of parameters also makes convolutional layers equivariant to translation [58]. The property of equivariance makes CNNs shift-invariant, which implies that they can recognise patterns or features in an image regardless of their spatial location. In other words, a shift-invariant CNN should be capable of identifying the same features or objects in different parts of the input without having to relearn the features for each possible location. The convolutional layer also has other essential parameters, such as stride and padding. Stride refers to the number of pixels skipped at a time (the same number of pixels skipped along the x-axis are also skipped along the y-axis). Padding refers to adding new entries to an array or matrix along its borders to enlarge the dimensions and obtain the desired dimensions.

2.3.2 Non-linearity layer

A non-linear activation function is applied after the convolutional layer in a CNN. The non-linear activation function adds non-linearity to the CNN to better account for the non-linear nature of patterns and features in images [57]. Section 2.2 (Figure 2.2) introduced some commonly used non-linear activation functions, with the ReLU activation function being the most popular. As a result, the non-linearity layer is often called the ReLU layer.

2.3.3 Feature pooling layer

Feature pooling layers serve two primary purposes. Firstly, it gradually decreases the input volume's spatial dimensions (width and height), which can effectively reduce the number of parameters and computations in the network [57]. Secondly, it helps manage overfitting, a common challenge in deep learning models. Feature pooling layers operate on each depth slice of the input independently, using either the max or average function. For example, when the average function is used, the pooling layer

computes the average values over a neighbourhood in each feature map. The neighbourhoods are traversed with a stride greater than one (yet smaller than or equal to the pooling neighbourhood).

2.3.4 Fully connected layer

After stacking multiple convolution blocks (convolution layer, non-linear activation layer, and feature pooling layer), fully connected (FC) layers (also known as dense layers) are typically used at the end of a CNN architecture to take the extracted features from the preceding convolutional block and use them to make predictions or perform classification tasks [57].

Convolutional Neural Networks (CNNs) can learn to detect edges and small structures in their lower-level layers by stacking multiple convolution blocks and using backpropagation [58]. These structures can then be used in higher-level layers to detect and classify more complex objects such as cars, dogs, and cats. Using convolutions, CNNs can extract meaningful features, which are then used to classify objects of interest. Much like humans use features such as colour and structure to distinguish objects, CNNs use features extracted through convolutions.

2.3.5 Batch normalisation layer

Ioffe and Szgedy first introduced batch Normalisation (BN) [59] in their 2015 paper titled "*Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.*" As the name suggests, the BN layer accelerates training by normalising the activations of a given input volume before sending it to the next layer in the network. Given a mini-batch d-dimensional input $X = [x^1 \cdots x^d]$ for a layer, the BN layer normalises each mini-batch activation in four steps:

- **Step 1:** The mean of the input layer is calculated using

$$\mu_{\mathrm{x}} = \frac{1}{d}\sum_{i=1}^{d} x^i. \tag{2.3}$$

- **Step 2:** The variance of the mini-batch is calculated using

$$\sigma^2_{\mathrm{x}} = \frac{1}{d}\sum_{i=1}^{d}(x^i - \mu_X)^2. \tag{2.4}$$

- **Step 3:** The mini-batch layer inputs are normalised using

$$\hat{X}_{\mathrm{i}} = \frac{(x_i - \mu_X)}{\sqrt{\sigma^2_X + \epsilon}}. \tag{2.5}$$

  To prevent the square root of zero, ϵ is added to the computation.

- **Step 4:** The normalised mini batch values are calculated using

$$y_{\mathrm{i}} = \gamma x_i + \beta. \tag{2.6}$$

  The values of γ and β are learned during training.

2.3.6 Dropout layer

The dropout (DO) layer is a regularisation technique aimed at improving testing accuracy while sacrificing training accuracy to prevent overfitting [57]. During training, for a given mini-batch, the DO layer alters the network architecture by randomly disconnecting inputs from the previous layer with a probability of p. Figure 2.5 illustrates the functioning of dropout layers with a dropout probability of 50%. The illustration depicts a halving of connections between layer 1 and layer 2 following the application of the dropout layer with $p = 0.5$. Once the forward and backward passes are computed, the dropped connections are restored, and another set of connections is randomly selected for dropout.

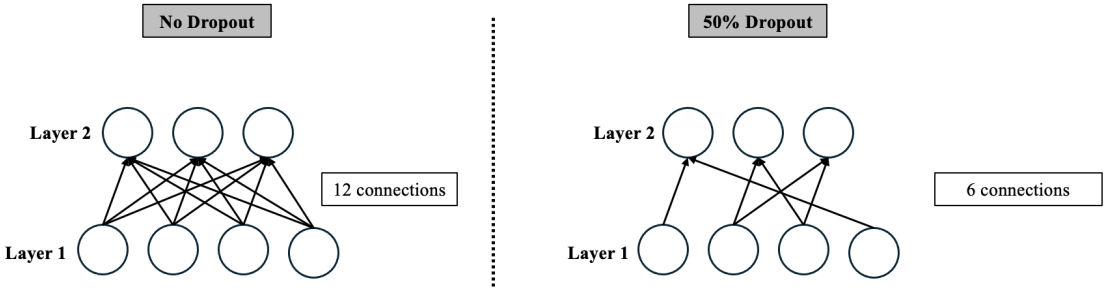


Figure 2.5: The concept of dropout layers. **Left**: Two layers of neurons that are fully connected. **Right**: Two layers of neurons with a dropout probability of 50%. *Image adapted from* [57].

2.4 Training deep learning algorithms

Section 2.3 discussed the process of building CNN architectures. This knowledge allows one to create a complete CNN architecture with trainable weights. Four essential components are required to train the weights: a dataset, a loss function, the CNN architecture itself, and an optimisation method. The four components can be summarised as follows:

- **Dataset:** The dataset is the first component in training a neural network – the data itself, along with the problem we are trying to solve, defines our end goals. A neural network learns to map a set of inputs to a set of outputs from the training dataset.
- **Loss function:** A loss function quantifies the performance of a model in classifying input data points in a dataset. Lower loss values indicate better classification accuracy. In other words, the loss function measures the effectiveness of the scoring function. The task often influences the selection of a loss function. For example, mean square error (MSE) is commonly used for regression tasks, binary cross-entropy loss (BCE) for binary classification problems, and categorical cross-entropy loss (CCE) for multi-class classification problems. Object detection tasks require a combination of multiple loss functions such as cross-entropy loss, smooth L1 loss, IoU loss, focal loss, and YOLO loss. For a more comprehensive review of deep learning loss functions, refer to [60] and [61].

- **Model:** The network architecture can be considered the first choice to make as one of the components to train a deep learning algorithm. The choice of the network architecture is often influenced by factors such as the number of data points in the dataset, the number of classes, how similar or dissimilar the classes are, and intra-class variance.

- **Optimisation method:** When training a deep learning network, the optimisation method is a crucial component. Optimisation algorithms are essential in training deep learning algorithms, allowing neural networks to learn patterns from data. These methods optimise a neural network by finding optimal weights to minimise the loss function. The most used method for optimising neural networks is gradient descent. The idea behind gradient descent is to evaluate parameters iteratively, compute the loss, and then move in a direction that will minimise the loss. Gradient descent can be implemented as shown in Algorithm 1 [62]. The gradient descent algorithm requires the weight matrix (θ), learning rate (α), the maximum number of iterations (max_iter), and the training data. For supervised learning, the training data represents a training set $\{x^{(1)}, \cdots, x^{(m)}\}$ with corresponding targets $y^{(i)}$.

  Gradient descent starts by computing the gradient (∇) of an objective function ($J(\theta)$) that represents the loss (L) in Eq. (2.7). In Eq. (2.7), L_{f} represents an arbitrary loss function and f represents a scoring function that maps the inputs $\{x^{(1)}, \cdots, x^{(m)}\}$ to outputs $y^{(i)}$. The loss represents the error of a neural network over the training examples and the weight matrix. The final step updates the weight matrix using the computed gradient ($\nabla J(\theta)$)) and the learning rate. The learning rate is an important parameter that controls the step size during iterations. The maximum number of iterations is also known as the number of training epochs.

$$L = \frac{1}{m} \sum_{i=1}^{m} L_{\text{f}}(f(x^{(i)}; \theta), y^{(i)}) \tag{2.7}$$

  Variants of the gradient descent method, such as stochastic gradient descent (SGD) and various gradient descent optimisation methods, such as Nesterov accelerated gradient and Adam, exist. The reader can refer to [63] for a comprehensive review.

Algorithm 1: Gradient descent

Data: Initial parameters θ, learning rate α, maximum number of iterations max_iter, m training examples

Result: Optimal parameters θ

Initialize iteration counter $iter \leftarrow 0$;

while $iter < max\_iter$ **do**

  Compute the gradient: $\nabla J(\theta) \leftarrow \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta L(f(x^{(i)}; \theta), y^{(i)})$;

  Update parameters: $\theta \leftarrow \theta - \alpha \nabla J(\theta)$;

  Increment iteration counter: $iter \leftarrow iter + 1$;

2.5 Generalisation and regularisation

In machine learning, the main goal is to develop a model that can effectively handle new and unseen input data, a concept known as generalisation. To achieve model generalisation, data for a specific problem is usually divided into two sets: a training set and a test set. The training set is utilised to train a machine learning model, while the test set measures the model's performance. Since the model's effectiveness in real-world applications is determined by its performance on the test set, it is often used as an indicator of its ability to generalise to new inputs.

Training a machine learning model involves minimising the errors made by the model on the training set. The combined errors of the model over the training set are known as the training error. However, it is also necessary to reduce the errors of the model on the test set, which is called the generalisation error. The generalisation error is estimated by measuring the model's performance on the test set. The generalisation error estimation assumes that the data examples in the training and test set are independent and that the training and test set are identically distributed, i.e., drawn from the same probability distribution.

The performance of a machine learning model depends on its ability to minimize both the training error and the gap between training and test error [64]. These two factors present the main challenges in machine learning: underfitting and overfitting. Underfitting occurs when a model cannot achieve low error values on the training set, while overfitting happens when the gap between the training error and test error is too large.

The model's capacity is what controls its tendency to overfit or underfit. Model capacity refers to the model's ability to fit a wide range of functions. In deep learning, increasing the number of layers in the network can increase the model's capacity, whereas removing layers and applying regularization techniques such as the dropout layer (Section 2.3.6) can reduce it. Figure 2.6 visually represents the typical relationship between overfitting and underfitting and model capacity.

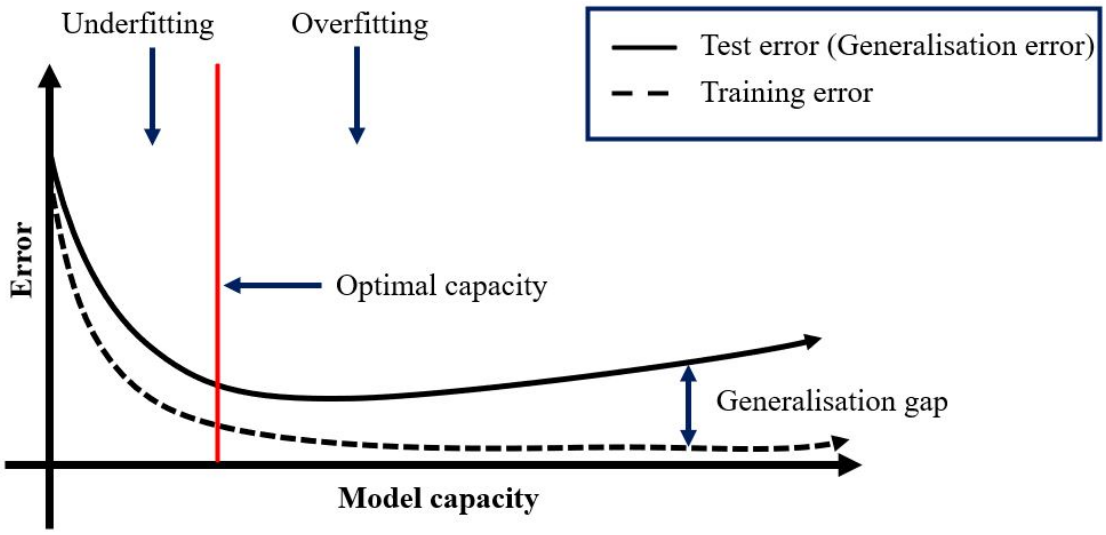


Figure 2.6: Typical relationship between model capacity and error. The red line separates optimal model capacity from **underfitting** (left end of the graph) and **overfitting** (right end of the graph). **Optimal model capacity** is reached when the generalisation and training error curves are level. As model capacity increases, the training error decreases but the generalisation gap increases. Ultimately, the model overfits to the data when the size of the generation gap outweighs the training error. Note: illustration adapted from Goodfellow *et al.*, page 113 [64].

Regularisation is a technique used to prevent overfitting in machine learning. It is crucial in deep learning as it helps models perform well on new data even when the training data is limited. According to Goodfellow *et al.* [64] regularisation is "*any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.*" However, in this work, the broader definition provided by Jan *et al.* [65] will be used. Regularisation in this context refers to a group of supplementary techniques that aim to improve a model's ability to generalise on a specific test set.

Consistent with the adopted definition, we can group the methods of regularisation into five categories [65]:

- **Regularisation via data:** This involves transforming the training data to create a new training set. It includes techniques such as feature extraction or preprocessing that alter the feature space or distribution of the data to a representation that simplifies the learning task
- **Regularisation via the network architecture:** This refers to selecting specific network architecture choices to fit the data well. This includes decisions on layer operation, noise, and model selection.
- **Regularisation via the error function:** This group is concerned with choosing an appropriate loss function based on the task and characteristics of the data.

- **Regularisation via the regularisation term:** The methods in this group modify the loss function by adding a regularisation term $\lambda R(W)$, where λ is a weighing term that controls the amount or strength of the regularisation, $R(W)$, being applied.
- **Regularisation via optimisation:** The last group includes the choice of techniques used to initialise and update neural network weights, along with stopping criteria for the optimisation procedure.

2.6 Object detection

Object detection is a task in computer vision that involves identifying and locating multiple objects within an image [58]. Two main types of deep learning-based object detection algorithms exist: two-stage and single-stage detectors. A two-stage detector starts by generating candidate regions that might contain an object before using a CNN architecture to extract features or classify those regions. A single-stage detector starts from an image input. Using a CNN architecture, it produces bounding boxes with corresponding class labels in a single forward pass without requiring region proposals. The R-CNN family [66]–[68] is the most well-known group of two-stage object detectors, whereas the YOLO family [36], [69]–[73] is the most popular group of single-stage object detectors.

This thesis focuses on object detection to improve the robustness of distracted driver detection. Specifically, the study uses the YOLOv7 [36] model to detect key human body parts involved when a driver operates a vehicle and classify their state into activities. YOLOv7 is a state-of-the-art object detector, and it has proven to be superior to other object detectors, such as Faster R-CNN [74] and EfficientDet [75], in terms of accuracy and inference speed. An overview of the YOLOv7 model's working principle will be introduced in Section 2.6.1. The details of the YOLOv7 architecture will be provided in Section 2.7.

2.6.1 YOLO working principle

YOLOv7 belongs to the You Only Look Once (YOLO) family of one-stage detectors, and it is the seventh version of the YOLO series. The first version of the YOLO family was presented in 2016 by Redmon *et al.* [69] in their ground-breaking paper titled "*You Only Look Once: Unified, Real-Time Object Detection*" (YOLOv1). The main idea behind YOLO algorithms is that the problem of object detection is framed as a regression problem, allowing them to generate object bounding boxes and corresponding class probabilities from image pixels in one forward pass through the network, as illustrated in Figure 2.7. This unified approach to object detection has led to the widespread use of YOLO algorithms due to faster inference speeds and high accuracy [76].

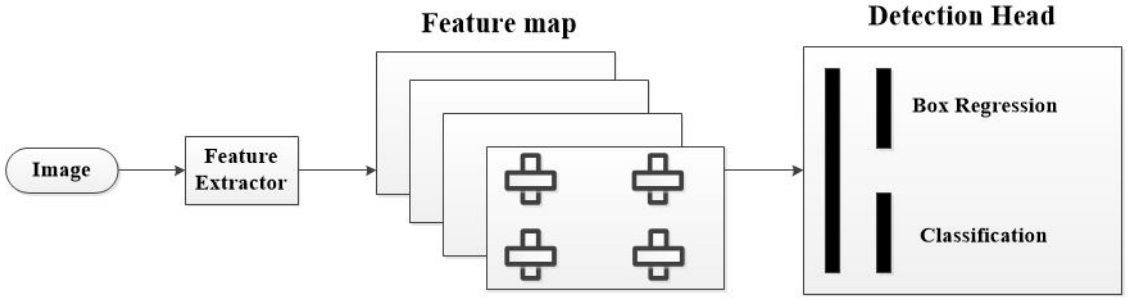


Figure 2.7: Processing pipeline of YOLO object detectors. The input image passes through a feature extractor that generates the feature map. Thereafter, the detection head is applied directly to the feature map to generate bounding box predictions and corresponding class probabilities. *Note: illustration inspired by PyimageSearch.*

As shown in Figure 2.8, the basic working principle of a YOLO algorithm involves dividing an input image into an $S \times S$ grid. A grid cell is responsible for detecting an object if the centre of the object falls on that grid cell. Each grid cell predicts B bounding boxes and the corresponding confidence scores. The confidence score indicates how confident the model is that there is an object in the box and how accurate it thinks the prediction is. The final detections are obtained by refining the bounding boxes using the non-maximum suppression (NMS) technique based on the confidence scores. If the confidence score of a bounding box is above some threshold, then the class with the highest probability is reported. If the confidence score is below the threshold, that box is ignored and assumed to belong to the background class.

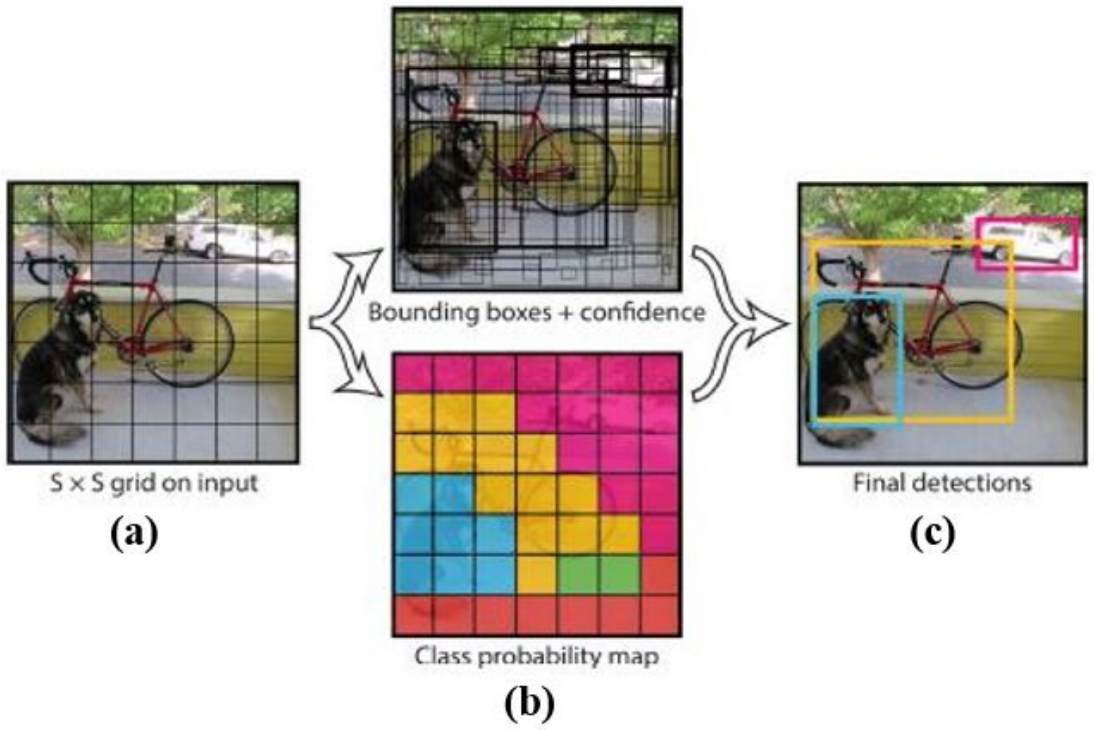


Figure 2.8: Working principle of the YOLO algorithm: **(a)** the input image is first divided into an S x S grid; **(b)** A grid cell is responsible for detecting an object if the centre of the object falls into it. Each grid cell is responsible for predicting B bounding boxes and the confidence score for each box; **(c)** Final detections are determined by refining the bounding boxes based on the confidence scores. *Note: image from* [69].

2.7 YOLOv7 architecture

Since the first YOLO model was introduced, researchers have published subsequent versions. Most iterations aim to improve speed and accuracy by using new loss functions, different backbone CNN architectures, introducing new concepts (anchor boxes, dense anchor boxes, and feature pyramid networks) and using different training routines. Table 2.1 shows the evolution of the YOLO family and the key architectural changes. The reader is referred to [77] for a review of the timeline and developments of the YOLO family of object detectors. The details of the YOLOv7 model are discussed below.

Table 2.1: YOLO architectural differences [78].

YOLO version	Backbone	Neck	Key changes and advancements
YOLOv1 [69]	CNN	None	The first one-stage object detector
YOLOv2 [70]	Darknet-19	Custom to detect small objects	Anchor boxes, batch normalisation, combined detection and classification
YOLOv3 [71]	Darknet-53	Feature Pyramid Network (FPN)	Darknet-53 backbone, FPN inspired design, multi-scale predictions
YOLOv4 [72]	CSPDarknet-53	Spatial pyramid pooling (SPP) +Path Aggregation Network (PANet)	SPP integration, PANet integration, Cross Stage Partial Network (CSPNet), Mish activation, data augmentation strategies
YOLOv5[1]	CSPDarknet-53	Spatial pyramid pooling fast (SPPF), New CSP-PAN	SPPF, augmentation methods, Model variants PyTorch integration.
YOLOv6 [73]	1. CSPStackRep Block (for large models) 2. RepBlock (for small models)	RepPAN	Bi-directional Concatenation (BiC) module, Anchor-Aided Training strategy, Enhanced backbone (EfficientRep) and neck design (RepPAN), and Self-distillation strategy
YOLOv7 [36]	Extended Efficient Layer Aggregation Network (E-ELAN)	CSPSPP+(ELAN, E-ELAN PAN	Introduced a new backbone (E-ELAN), model scaling techniques, and training bag of freebies

The focus of YOLOv7 is on improving object detection accuracy by using efficient modules and optimization methods while maintaining inference speed [36]. The YOLOv7 authors called these modules and optimisation methods "trainable bag-of-freebies". The bag of freebies concept was first introduced in YOLOv4 and refers to techniques that aim to improve detection accuracy without reducing the inference speed. The key changes and advancements in YOLOv7 are: (1) architecture optimisation; and (2) training process optimisation. These key changes are discussed below.

[1]No official research paper. Documentation can be found at https://docs.ultralytics.com/yolov5

2.7.1 Architecture optimisation

The key architectural changes in YOLOv7 include the introduction of the extended efficient layer aggregation network (E-ELAN) and model scaling for concatenation-based models. E-ELAN is the core innovation in the backbone of the YOLOv7 model and builds on previous research that focused on the path of maximal layer efficiency with Cross Stage Partial Networks (CSP) while considering the memory usage of layers along with the distance it takes a gradient to back-propagate through the layers. Specifically, the authors of YOLOv7 extended ELAN by using group convolution, feature map shuffling and merging cardinal features. This allows E-ELAN to effectively aggregate feature maps from different layers and scales, continuously augmenting the learning ability of the model which in turn improves the object detection accuracy of YOLOv7. Figure 2.9 shows the architectures of ELAN and E-ELAN. The key features of the E-ELAN backbone are:

- **Group convolution:** Group convolution is a variation of the standard convolution where instead of applying each filter on all channels of the input feature map, the channels of the input feature map are split into predefined groups and each filter within a group is applied to a specific set of channels of the input feature map [79]. Group convolutions allow models to learn features independently across the groups, allowing the model to learn diverse features in the data.
- **Shuffling:** Shuffling involves intermixing the information within the expanded feature maps that were created in the previous step (expand cardinality). Shuffling allows the YOLOv7 network to learn more complex feature interactions and dependencies that might not be apparent in individual groups.
- **Merging:** After shuffling the feature maps from all the groups, merge cardinality integrates information from diverse channels and groups, thereby allowing a deeper understanding of the image content for object detection.

Model scaling is a technique used to adjust the properties of a model's architecture to meet the needs (speed, accuracy, and resource consumption) of different applications [36]. Model scaling can optimise the model's width (number of channels), depth (number of layers), and input resolution. These factors can be adjusted to scale the model, thereby fitting the specific needs of an application at hand.

In YOLOv7, a compound scaling method for concatenation-based models is proposed. Unlike traditional methods with concatenation-based models which do not allow an independent analysis of the impact of different scaling factors [36], YOLOv7 employs a compound model scaling approach that maintains the initial design properties of a model and its optimal structure. This is achieved by coherently scaling the width and depth within computational blocks. For example, after scaling the depth of a computational block, the change of the block's output channel is also calculated. Then, the width on the transition layers is scaled by the same level of change. Essentially, YOLOv7 maintains efficiency by separating depth scaling (in the computational blocks) from width scaling (in transition layers). The compound scaling allows the generation of YOLOv7 models with varying accuracy and speed trade-offs.

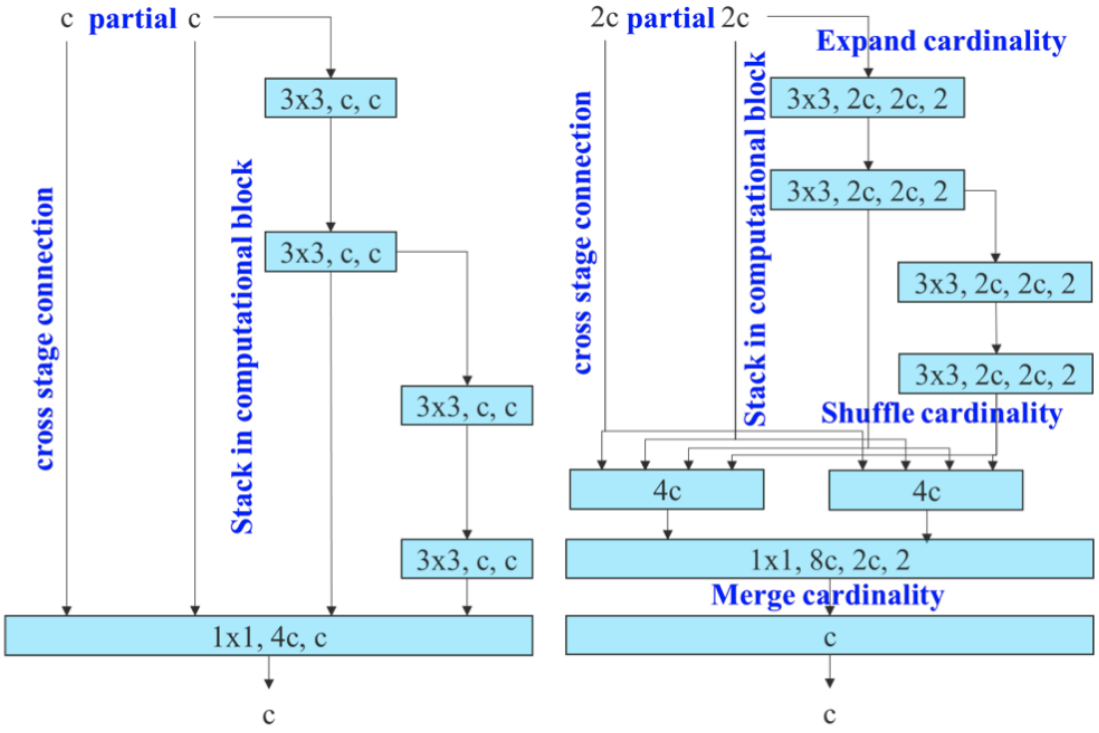


Figure 2.9: Extended efficient layer aggregation networks. **Left**: Structure of ELAN; **Right**: Structure of the extended ELAN (E-ELAN) used in YOLOv7. For each layer, E-ELAN expands the feature maps (expand cardinality), shuffles the feature maps (shuffle cardinality), and combines the feature maps (merge cardinality). All this is done without destroying the original gradient path. These three key features allow the YOLOv7 to improve the learned features and the training process. Image obtained from [36].

2.7.2 Training process optimisation

YOLOv7 introduced the following key trainable bag of freebies:

- **Planned re-parameterised convolution:** Re-parameterisation involves averaging the weights of a convolutional layer across different training runs [80] to create a model that is more robust to features it trying to learn. Essentially, this allows a model to leverage knowledge learned from different training scenario settings. In YOLOv7, a planned re-parameterized convolution method that uses RepConv [81] without identity connection is employed.
- **Coarse for auxiliary and for lead loss:** As mentioned in Section 2.6.1, a YOLO model has a backbone, neck, and head. The head in the YOLO framework is the final component responsible for generating final predictions using the features extracted by the backbone and processed by the neck. Inspired by Deep supervision [82], a technique commonly used for training deep learning networks, YOLOv7 uses multiple heads for prediction. The head

overseeing the final output is termed the lead head, while the auxiliary head, positioned within the middle layers, assists during training. The use of the auxiliary head allows YOLOv7 to have direct supervision of the detection head in the middle layers, rather than the standard approach where supervision is only at the output layer.

In addition, to improve the training process, YOLOv7 introduced a label assigner mechanism that assigns soft labels after considering the network prediction results with ground truth. Contrary to conventional label assignment, dependable soft labels employ calculation and optimization techniques that consider the quality and distribution of prediction output, as well as the ground truth, in contrast to conventional label assignment, which uses only ground truth to generate hard labels following predefined rules.

2.8 YOLOv7 model training

The implementation process to customise a YOLOv7 model begins with problem scoping. Problem scoping involves defining the identified problem and a performance metric that must be optimised, along with a satisficing metric. A satisficing metric is a metric that should be monitored to ensure that it does not exceed a predefined threshold value, but it does not necessarily need to be optimised.

The YOLOv7 model can be customised in three steps: set up, train, and verify. YOLOv7 model setup entails installing the YOLOv7 model code and its dependencies, preprocessing the data, selecting a version of the YOLOv7 model series, and selecting hyperparameters. In the second step, the model is trained to obtain weights that minimise training and test errors. The final step involves evaluating the model's performance and comparing it to the predefined value of the optimising metric. If the model's performance is satisfactory, it will be used for the final implementation. If not, steps one and two will be repeated until satisfactory model performance is achieved. Figure 2.10 illustrates the three-step development process to customise the YOLOv7 model.

The specific details about the implementation and activities involved in each step of the development process in Figure 2.10 are provided in Chapter 5.

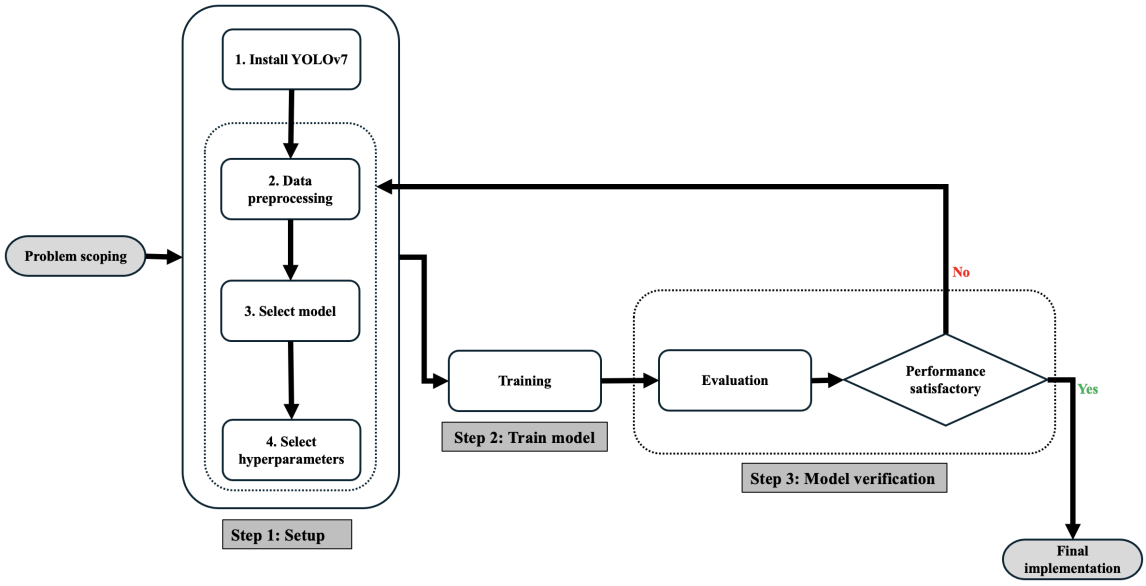


Figure 2.10: Overall implementation process to train and fine-tune the YOLOv7 model to custom data.

2.8.1 Object detection performance metrics

The common metric used to measure the accuracy of detections in object detection is average precision (AP) [83]. Precision measures the ability of an object detector to identify relevant objects. In object detection, precision is measured is measured using the intersection over union (Iou). IoU is a measurement based on the Jaccard Index and measures the similarity between two sets of data [84]. In object detection, IoU measures the overlapping area between a predicted bounding box B_p and the corresponding ground-truth bounding box B_{gt} and it is calculated as

$$J(B_p, B_{gt}) = IoU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})}. \tag{2.8}$$

Figure 2.11 illustrates how IoU is calculated using an example of detecting a stop sign. The value of IoU is a ratio and is normally between 0 and 1. An IoU score > 0.5 is normally considered a "good" prediction [85].

Using the IoU, the precision of an object detector for a specific class (AP) and the precision for all classes (mean average precision) in a dataset can be calculated. As a result, AP for a specific class is calculated by summing the IoU of all classes belonging to that class and dividing by the total number of data points in the class. Mean average precision (mAP) is calculated by averaging AP per class for all classes in the dataset. Typically, object detection results are reported using $mAP@0.5$, indicating that for an object in the test set to be marked as a "positive detection" it must at least have an IoU ≥ 0.5.

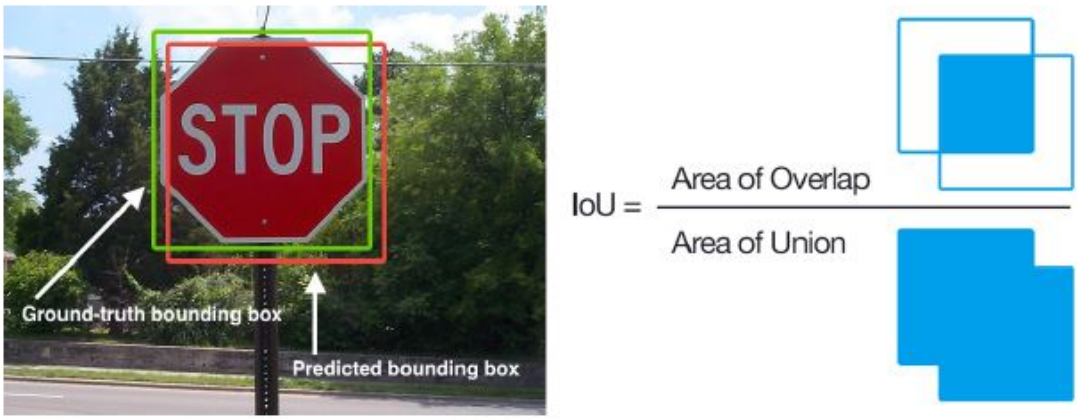


Figure 2.11: An example of calculating the IoU between a predicted and a ground-truth bounding box for detecting a stop sign [85].

2.9 Summary

This chapter laid the groundwork for the upcoming chapters. It introduced crucial fundamental concepts related to detecting distracted drivers, artificial neural networks, convolutional neural networks, training deep learning algorithms, the significance of model generalization and regularization in machine learning, and object detection. In the following chapters, the concepts introduced in this chapter will be further developed to address the research objectives outlined in Chapter 1.

Chapter 3 will provide the relevant literature that places the work done in this thesis within the existing body of knowledge.

Chapter 3

Literature review

The problem of detecting distracted drivers has been extensively studied in the literature, resulting in the proposal of various measurement methods. Section 1.2 presented an overview of these existing methods. This chapter aims to provide a detailed literature review of the existing methods. This review will support the first research objective (ROJ1) outlined in Section 1.4 of the thesis.

Following METHOD1 introduced in Section 1.5, the literature review focuses on three areas: deep learning-based distracted driver detection, robust distracted driver detection, and cross-dataset performance evaluation. The literature reviewed is obtained from academic databases such as IEEE Xplore, ScienceDirect, and Google Scholar. The focus of the thesis on deep learning-based methods is motivated by the superiority of deep learning algorithms over traditional algorithms for the task of distracted driver detection [40].

This chapter introduces common datasets used to train and evaluate deep learning-based distracted driver detection (Section 3.1). Section 3.2 provides a literature review on deep learning-based distracted driver detection methods. The chapter concludes by providing a literature review on the cross-dataset performance evaluation of distracted driver detection in Section 3.3.

3.1 Datasets

Datasets are crucial for effectively applying deep learning to real-world problems because deep learning algorithms learn patterns from training dataset features. This applies to detecting distracted drivers as well.

3.1.1 Public datasets

The first dataset in the area of driving behaviour analysis and distracted driving was introduced by Zhao *et al.* [86], [87]. The dataset is known as the Southeast University Driving-posture Dataset (SEU dataset). This dataset has side-view images of the driver performing four activities: (i) grasping the steering wheel, (ii) operating the shift lever, (iii) eating a cake, and (iv) talking on a cellular phone, see Figure 3.1. However, the dataset is not publicly available. All the papers ([13],

[88], [89]) that benchmarked using the dataset are affiliated with either Southeast University, Xi'an Jiaotong-Liverpool University, or Liverpool University, and they have at least one shared author [15].

Figure 3.1: Example images from the SEU dataset: **(a)** grasping the steering wheel; **(b)** operating the shift lever; **(c)** eating a cake; and **(d)** talking on a cellular phone (source: [88]).

Later, the State Farm Insurance Company released a dataset to find out if computer vision can spot distracted drivers. The insurance company held a competition named State Farm Distracted Driver Detection [90] on Kaggle, referred to as STF in this thesis. The State Farm dataset consists of 2D dashboard camera images showing ten different driving postures, as shown in Figure 3.2. However, despite the State Farm dataset being public, it was only limited to the purpose of the State Farm Distracted Driver Detection competition.

Due to the lack of a quality dataset, Billah *et al.* [91] created a four-class distracted driver dataset called the EEE BUET Distracted Driving dataset. The dataset was created using a Sony Cyber Shot 14.1-megapixel camera affixed on the front windscreen facing the driver inside the vehicles. The four distracted driving activities in the dataset include talking on the cell phone, texting on a cell phone, eating, and operating cabinet equipment. A total of 13 participants took part in the development of the dataset.

Figure 3.2: Representative samples images from the State Farm dataset.

Inspired by the State Farm dataset, Eraqi *et al.* [15] created a similar dataset called the AUC Distracted Driver Dataset, referred to as AUC2 in this thesis. The dataset was made public subject to signing an agreement form. A two-phase data collection method was followed — in the first phase, the ASUS ZenFone smartphone (Model ZD551KL) rear camera was used, and the DS325 Sony DepthSense camera was used in the second phase. In the project, 44 drivers from 7 countries were involved, of which 29 were males and 15 were females. However, it has been reported that the AUC dataset is not balanced. For example, the "reach behind" class only represents 7% of the complete data points [27].

In contrast, the "safe driving" class represents 21% of the complete dataset. In addition, not all drivers participated in all distraction activities. To remedy the shortcomings of the AUC dataset, Ezzouhri *et al.* [27] introduced a distracted driver detection dataset with 9 participants, referred to as EZZ2021 in this thesis.

Table 3.1 presents an overview of the three commonly used distracted driver detection image datasets, each featuring ten distinct classes. The table provides information regarding the environment in which these datasets were created (real or synthetic), the types of distractions captured, the number of drivers involved, and the overall dataset sizes.

Table 3.1: Ten-class commonly used distracted driver detection image datasets. The highlighted dataset names have links to the datasets.

Image dataset	Year	Environment	Type of distractions	Participants	Image samples
EZZ2021 [27]	2021	Real	1 safe driving, 9 distracted activities	9	29.2k
STF [90]	2016	Real	1 safe driving, 9 distracted activities	44	32.7k
AUC2 [15]	2019	Real	1 safe driving, 9 distracted activities	26	22.4k

Figure 3.4 (a) shows the distribution of images per class in the EZZ2021, STF, and AUC2 training datasets. It is evident from the image distribution that the EZZ2021 and STF training datasets are relatively balanced, with an almost equal number of images per class. On the other hand, the AUC2 training dataset is imbalanced. For instance, the "text left" class (C3) has only 641 images, while the "safe driving" class has 2346 images. The AUC2 training dataset's imbalance will likely impact the performance of algorithms trained on it. The reader can find the raw data showing the images' distribution per class in Appendix A of this work.

3.1.2 CSIR test dataset

The CSIR created a test dataset comprising a total of 510 images. The dataset's creation involved engaging five drivers, each operating different vehicles, and requesting them to perform each of the ten driver activities in the EZZ2021, STF, and AUC2 datasets. Video footage of the drivers performing these activities was captured using a GoPro camera, either affixed to an armband on the car's roof or handheld in certain instances. The images were captured in daylight to be consistent with other datasets.

Figure 3.3 displays sample images from the CSIR test dataset, while Figure 3.4 (b) presents the distribution of images per class within this dataset. Notably, the CSIR test dataset contains a smaller number of images in comparison to the EZZ2021, STF, and AUC2 test datasets. Additionally, it is noteworthy that the CSIR test dataset lacks images corresponding to the "drinking" (C6) and "make-up" (C8) classes.

Figure 3.3: Sample images from the CSIR distracted driver detection dataset.

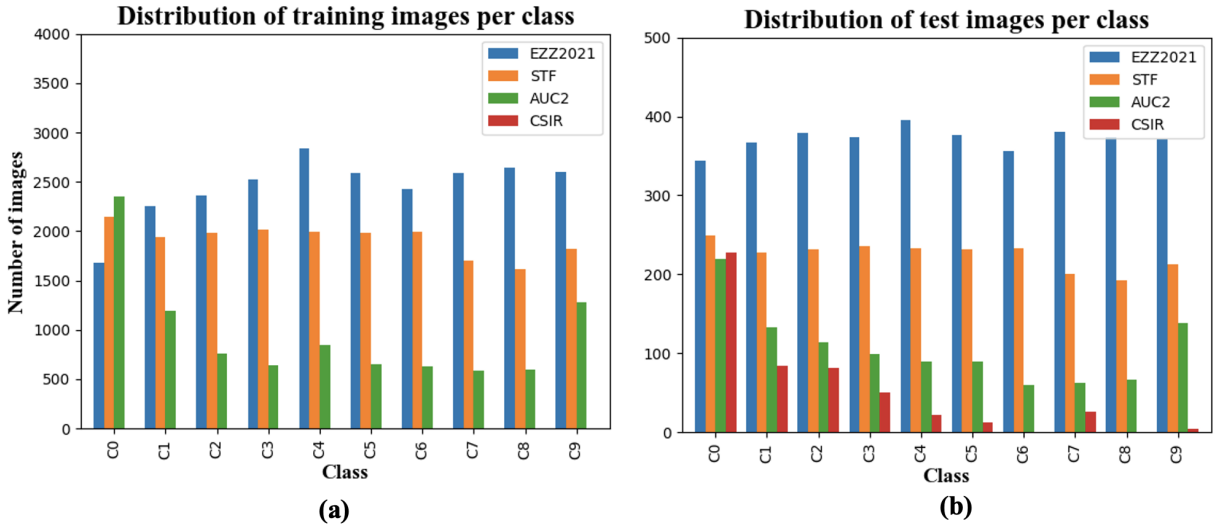


Figure 3.4: Distribution of images per class in the public and CSIR datasets. **(a)**: Training datasets. **(b)**: Testing datasets.

3.2 Deep learning in distracted driver detection

Given a particular deep learning recognition task, there are two basic ways to improve the performance of a model: improve the model or improve the input training data [33]. The first

solution involves changing or modifying the model to enhance its learning ability. The second solution involves improving features, representation of the object of interest, or simply increasing the amount of training data. Increasing the amount of training data poses two problems: (1) a considerable amount of data is required to achieve notable performance improvement [33]; (2) acquiring large and diverse in-car driver poses for distracted driver detection is a challenge due to the reluctance of drivers to compromise their privacy and the cost associated with using multiple devices for data collection, not to mention the labour-intensive annotation process [34]. The challenges of increasing the training dataset also make the apparent solution of fine-tuning a model to a task-specific dataset less attractive [35].

This work takes the approach of improving the features and the representation of the object of interest. In particular, the focus is on improving the representations of the driver. The work related to the proposed approach will be presented below.

Figure 3.5 shows an overview of major themes that emerge from deep learning distracted driver detection. These themes encapsulate the core approaches and methodologies used in deep learning distracted driver detection. The themes can be broadly categorized into two groups. The first group, "CNN feature classification", encompasses methods that use CNNs for feature extraction and driver behaviour classification. The second group represents ongoing efforts for robust distracted driver detection called "Robust Detection".

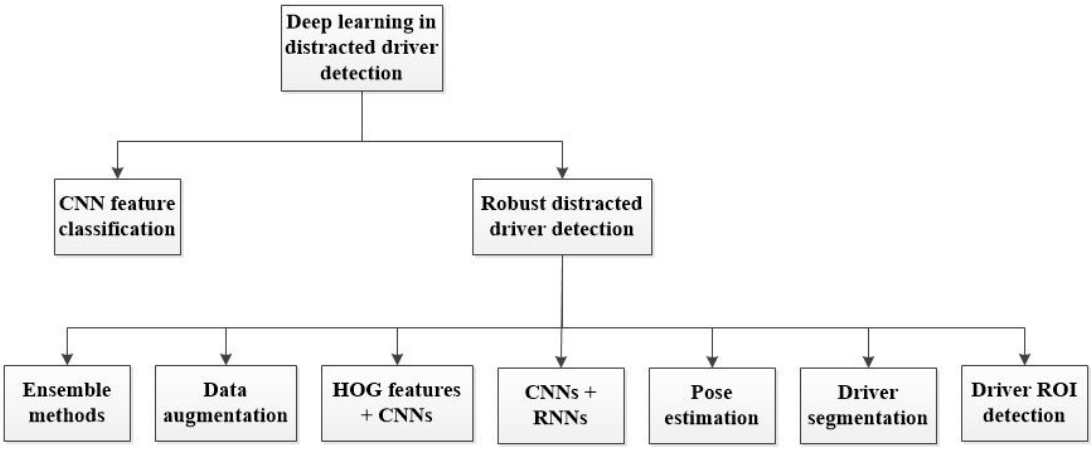


Figure 3.5: Themes in deep learning distracted driver detection.

3.2.1 CNN feature classification-based methods

In the beginning, deep learning-based distracted driver detection methods involved using CNNs for feature extraction and subsequently used a fully connected multilayer perceptron (MLP) [12], [13], as illustrated in Figure 3.6. These methods were trained using the classical machine learning framework for supervised learning algorithms.

In the classical training framework, a model is trained for a specific task and domain, assuming that there is enough labelled data for the same task and domain. The model is expected to perform well on unseen data from the same task and domain. However, when presented with data from a different task or domain, new labeled data is needed to train a new model that is expected to perform well on unseen data from that task or domain. The process of training a supervised machine learning model in a traditional learning setup is illustrated in Figure 3.7.

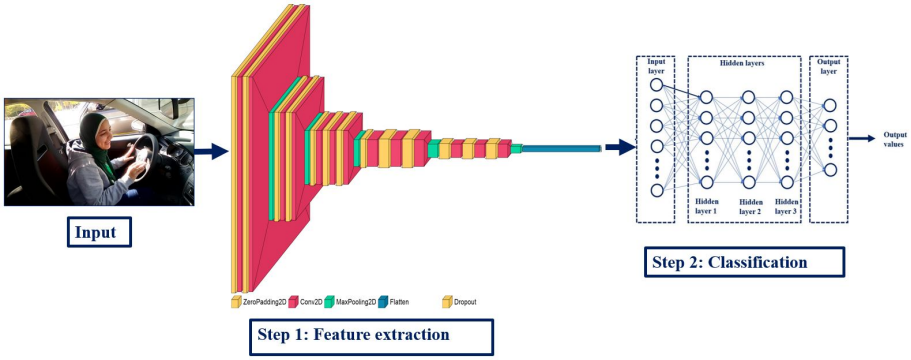


Figure 3.6: Typical framework used by CNN feature extraction methods.

Even though distracted driver detection algorithms implemented using the supervised learning setup shown in Figure 3.7 have demonstrated commendable success, a major limitation is that a large amount of labelled data is required [14]. To remedy this, researchers have used transfer learning. Transfer learning reduces the need for a large amount of labelled data by leveraging stored knowledge learned from a previous related task or domain, known as the source task and source domain. The stored knowledge gained from training a model on the source domain is applied to the target task and target domain as illustrated in Figure 3.8.

In the case of distracted driver detection, transfer learning approaches typically use CNN architectures pre-trained on large computer vision image datasets like ImageNet [14], [92]. These approaches replace the MLP fully connected (FC) layers of the pre-trained network with new FC layers and fine-tune them to recognize specific classes in distracted driver detection. Some of the commonly used pre-trained models for this task include AlexNet, VGG16, VGG19, ResNet50, InceptionV3, Xception, and DenseNet. The reader is referred to [14] for a more detailed evaluation and comparison of transfer learning approaches for distracted driver detection.

While transfer learning can often achieve high accuracy on a single dataset, the performance is limited to intra-dataset only. This is mainly due to overfitting caused by the limited diversity within current distracted driver detection datasets [38]. Current datasets mainly come from experiments in simulators or real car environments [17], which produce images with similar backgrounds and within a narrow range of distracted driving scenarios. As a result, the generalisation gap of the model (Section 2.5) becomes too large and results in poor model generalisation.

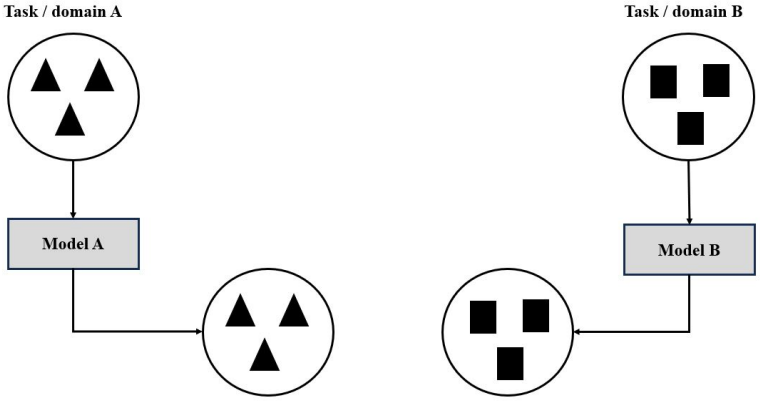


Figure 3.7: Traditional setup for training a supervised machine learning model. Model **A** trained on a specific task or domain **A** is expected to perform well on unseen data from the same domain. When given data from another domain or task **B**, new labelled data from the same task or domain will be required again. The new data can be used to train a new model **B** that is expected to perform well on new unseen data from task or domain **B**. *Note: figure adapted from* [93].

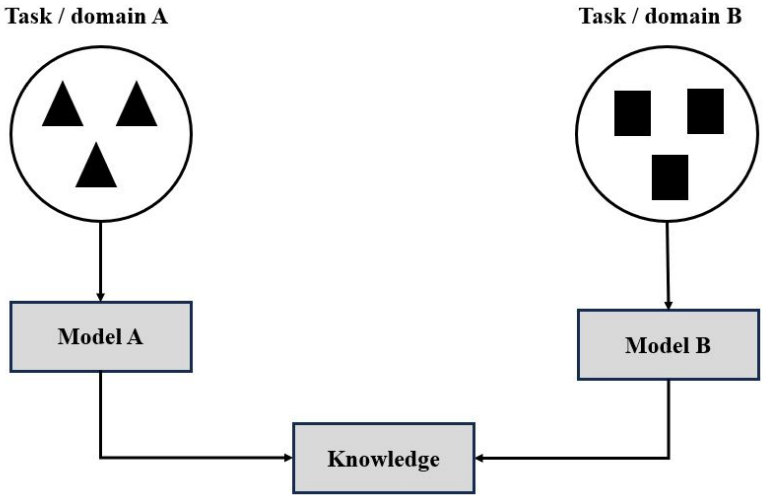


Figure 3.8: Transfer learning setup for supervised machine learning. In transfer learning, stored knowledge that is learned from a previous related task or domain **A**, known as the source task and source domain is used to train a model **B** on the source domain and is applied to the target task and target domain **B**. *Note: figure adapted from* [93].

3.2.2 Robust distracted driver detection methods

Due to the overfitting issue of distracted driver detection algorithms trained using transfer learning, researchers have proposed various methods to improve the robustness of deep learning-based methods. Table 3.2 summarises methods that aim to improve the generalisation of deep learning-based distracted driver detection. These methods are reviewed in detail below.

Table 3.2: Methods that focus on improving the robustness of deep learning-based distracted driver detection.

Theme	Approach	References
Ensemble methods	These methods leverage the characteristics of different CNN architectures by combining multiple CNN models to enhance accuracy and robustness.	[15], [16]
Data augmentation methods	The methods in this group use advanced data augmentation techniques to enhance the training dataset's diversity and balance.	[17], [18]
CNN features + HOG features	Distracted driver detection using HOG feature extraction captures shape and edge information from images. It improves the algorithm's ability to classify behaviours based on unique image patterns.	[5], [19], [20]
CNNs + RNNs	These methods combine CNN and RNN architectures to leverage the strengths of both architectures. CNNs are effective at feature extraction from images, while RNNs handle sequential data and context modelling. This fusion enhances the algorithm's ability to capture temporal dependencies in the data, allowing for more accurate and context-aware detection of distracted driving behaviours.	[21], [22]
3D CNNs	These methods use 3D CNN architectures instead of 2D CNNs to consider both spatial and temporal information. The use of 3D CNNs enhances the algorithm's capacity to detect and classify distracted driver actions in a video context, providing a more comprehensive and accurate approach to the task.	[23], [24]
Pose estimation	Use features derived from human key points detected using pose estimation algorithms. The pose estimation features are often combined with CNN features for driver behaviour classification.	[25], [26]
Driver segmentation	Remove background noise by using instance segmentation algorithms to separate pixels belonging to the driver from the background pixels. The segmented images are then used to train a CNN architecture.	[27], [28]
Driver ROI detection	Remove background noise by using instance segmentation algorithms to separate pixels belonging to the driver from the background pixels. The segmented images are then used to train a CNN architecture.	[9], [29]–[31]

Ensemble methods are based on the concept of bootstrap aggregating [94], which is a technique that reduces generalisation error by combining several methods. Bootstrap aggregation involves training several models separately and subsequently having all the models vote on the output of examples in the test set. As a result, methods that use the ensemble approach take advantage of different CNN architectures that capture diverse and complementary features from the input data. By combining multiple architectures, an ensemble method can potentially represent a broader range of features and patterns, enhancing the model's ability to recognize different aspects of the data.

For example, Abouelnaga *et al.* [15] introduced a genetically weighted ensemble of CNNs trained on five different image sources: raw images, skin-segmented images, face images, hands images, and "face+hands" images. The study used four pre-trained deep learning models (AlexNet, Inception-V, ResNet50, and VGG-network) and fine-tuned them for driver distraction identification.

While ensemble methods have been reported to be successful in mitigating overfitting, other researchers have raised concerns regarding the computational complexity and cost of training multiple CNN models like VGG16, AlexNet, ResNet50, and Inception-V3 [22], particularly in real-time inference that is crucial for autonomous driving applications.

As an alternative, data augmentation-based techniques have been used to mitigate the issue of overfitting. Data augmentation-based approaches are based on the premise that increasing the diversity of the training dataset will improve the generalisability of the model (Section 2.5). Ou *et al.* [17] proposed an advanced data augmentation-based approach that employs GANs to generate synthetic data, thereby increasing the diversity of the training dataset. The authors claim they collected a diverse dataset of drivers in different driving conditions and activity patterns from the Internet and trained generative models for multiple driving scenarios. By sampling from these generative models, they augmented the collected dataset with new training samples and trained a CNN model for distracted driver detection. However, some researchers have reported difficulties in training GANs [35]. In addition, synthesised images often suffer from poor image quality [34].

To reduce overfitting, Jaco *et al.* [18] presented a class-based data augmentation approach that first detects four important key points, i.e., driver head, left hand, right hand, and the steering wheel. Regions of the detected key points are extracted and combined to form a single image. Thereafter, a class-based data augmentation that randomly interchanges regions between different images of the same class is applied. During training, the authors noticed that the training error decreased slower compared to the validation error when compared to other approaches that use the full image for training and validation. This trend suggests that the network could generalise well compared to the network that used the full image.

Researchers have also proposed methods that use the histogram of oriented gradients (HOG) feature descriptor. The HOG feature descriptor was first proposed for human detection and captures the human shape in an unambiguous way [95], allowing it to detect humans under different conditions. HOG has two major advantages. First, it captures the characteristics of an object by capturing the edges or gradient structure of the local shape. Second, it captures the local representation of an object with an easily controllable degree of invariance to local geometric and photometric transformations. As a result, if translations or rotations are smaller than the local

spatial or orientation bin size, only a small difference is observed.

Distracted driver detection algorithms use the HOG feature descriptor to remove background noise and only capture the outline of the driver's posture [19] (see Figure 3.9). Some researchers train a CNN model on the HOG feature images [19] or combine the HOG features with CNN features [5], [20] to improve model generalisation.

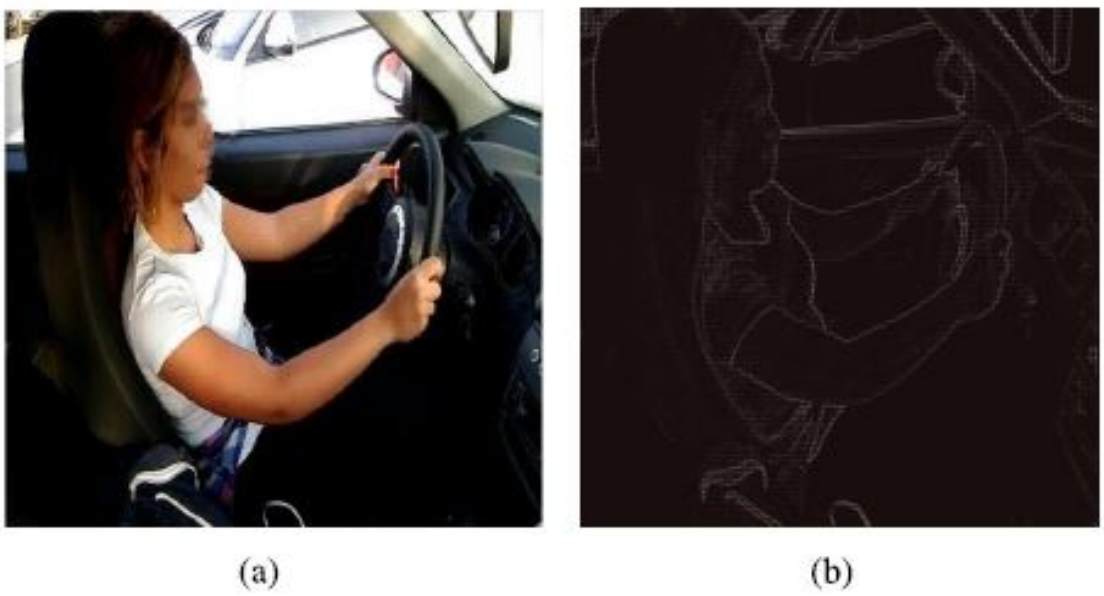


Figure 3.9: Original Raw images **(a)** are pre-processed to produce the HOG feature image **(b)** which does not have background noise [19].

Similar to the methods that use the HOG feature descriptor to remove background noise, some researchers have proposed methods that use human key points detected using a pose estimation algorithm [25], [26]. A pose estimation algorithm estimates the coordinates of human key points such as nose, mouth, hands, and shoulders. Common pose estimation algorithms used in distracted driver detection include High Resolution Net (HRNet) [96], and OpenPose [97]. Pose estimation-based algorithms extract the driver's human key points and use the posture information to reduce background noise and variation due to viewpoint. The driver posture information is often used in conjunction with CNN features [26], [97] or used directly to train shallow learning algorithms such as random forest [25]. Figure 3.10 shows sample outputs of a pose estimation-based method that involved training a random forest algorithm on human key point information extracted by the OpenPose algorithm.

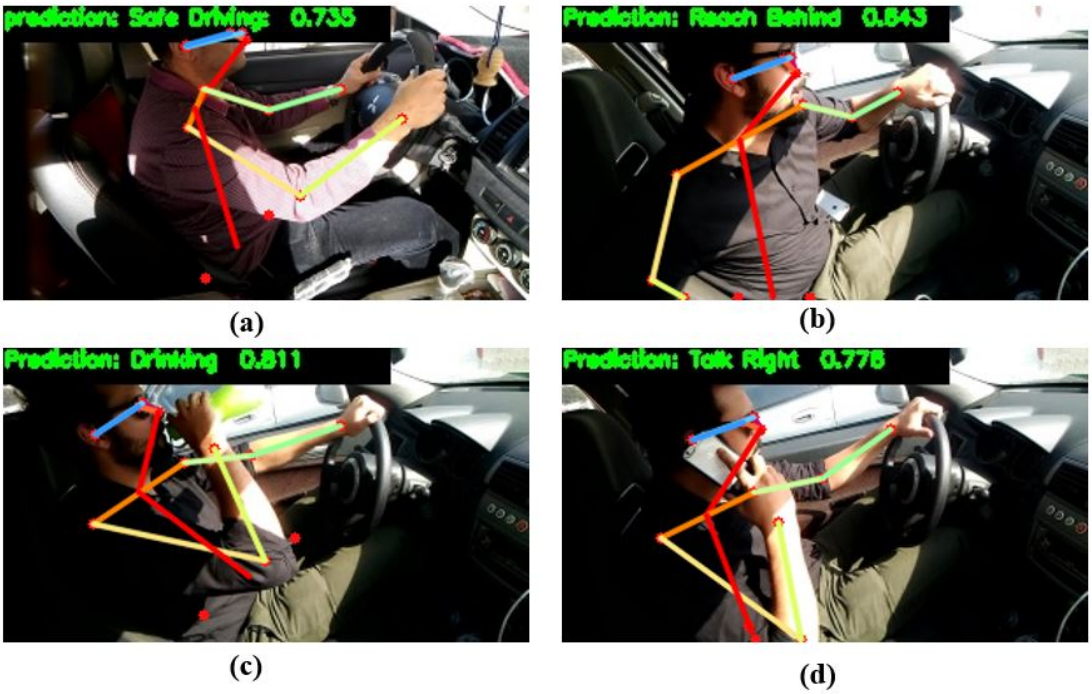


Figure 3.10: Sample outputs from a pose estimation-based method: **(a)** safe driving class predicted; **(b)** reach behind class predicted; **(c)** drinking class predicted; and **(d)** talk right class predicted.

The methods presented so far in this thesis use static images to train and test deep learning distracted driver methods. The major limitation of using static images is that it disregards temporal information [23]. Researchers have reported that CNN-based distracted driver detection models struggle to classify some distractions due to their spatial similarities with other postures. They claim that the only way to accurately classify such postures is by considering their spectral features, which provide more information about the images. To remedy this issue, researchers have proposed hybrid approaches that use CNNs to extract spatial features and recurrent neural networks (RNN) such as bidirectional Long Short-Term Memory networks to extract spectral features (temporal information) [22]. RNNs are neural networks that are specialised for processing sequential data [98]. Other hybrid CNN-RNN distracted driver detection methods use CNN features to extract spatial features and also use LSTMs to analyse vehicle data [21], [47]. Subsequently, the features from the two networks are typically forwarded to a Bayesian network for classification. However, the focus of this thesis is on improving the cross-dataset performance of CNN-based distracted driver detection.

In addition to the above approaches based on removing background noise, several methods that focus on using specific regions of the image rather than the full image have been proposed. One prominent technique involves employing instance segmentation algorithms to eliminate background noise and restrict CNN models to learning features solely from the driver [27], [28], [99]. For instance, Ezzouhri *et al.* [27] presented a CNN-based approach that employs a human segmentation

algorithm called Cross-Domain Complementary Learning (CDCL) to preprocess RGB images. The resulting pre-processed dataset is then utilized for training a CNN architecture. By leveraging instance segmentation, these methods aim to enhance the model's focus on driver-related features while reducing the influence of irrelevant background information.

Furthermore, to enhance the robustness of distracted driver detection, researchers have explored the use of object detection techniques to identify specific regions and objects [9], [12], [29]–[31], [100], which aligns with our proposed approach. Yan *et at.* [12] proposed a vision-based approach that used a modified R*CNN framework with two input regions: the primary region encompassing the entire driver image and the secondary region consisting of skin-like regions extracted using a Gaussian Mixture Model (GMM). The two regions were then forwarded to a deep convolutional neural network called R*CNN to generate driver action labels. However, the approach is not an end-to-end deep learning model as it requires region proposals generated by the GMM model and subsequent classification. Such complex pipelines are slow and challenging to optimize since each model needs separate training [69]. Additionally, the cross-dataset performance of the algorithm was not evaluated.

Another study by Le *et al.* [100] proposed a Multiple Scale Faster-RCNN approach that employed a standard Region Proposal Network (RPN) for generating region proposals. The approach incorporated feature maps from shallower convolutional layers for ROI pooling and aimed to detect individual objects such as hands, cell phones, and steering wheels. However, this study solely focused on cell phone distraction and neglected the driver's attention to the road. In addition, the Multiple Scale Faster-RCNN model was only trained and tested on a custom dataset created by the authors. The ability of the model to generalise on new data remains unknown. Similarly, another related study [31] introduced a distracted driver detection method primarily centred around cell phone detection, overlooking other aspects.

A distracted driver detection method very similar to the approach proposed in this thesis was presented by Sajid *et al.* [29]. The proposed method uses the EfficientDet model [75] to detect distraction objects and the ROI of the driver body parts and use an EfficientNet model [101] for classification. In summary, the approach involves the following steps: extract CNN features on an input image, classify the image, detect distraction objects and the ROI of the driver body parts, and finally combine the classification label with the detection label to obtain the final prediction. The approach combines both image classification and object detection. There is one major difference between this method and the approach proposed in this study. Instead of combining the full image and the driver ROI for driver behaviour recognition, the approach proposed in this thesis uses the Yolov7 model to detect driver body parts and classify their state into different activities in one forward pass. The final prediction is made by evaluating two conditions: "eyes on the road" and "both hands on the steering wheel." In addition, the cross-dataset performance of the proposed approach is evaluated on three distinct distracted driver detection image datasets. Further, the performance of the proposed approach is evaluated on a custom dataset to assess its potential for deployment in the real world.

3.3 Cross-dataset performance evaluation

Deep learning models typically perform well when the distribution of new data is similar to the training data, [64]. However, when the input data's distribution differs from the training data, deep learning models may perform poorly, a concept known as data shift or domain shift [102], [103]. The issue of domain shift has created the need to evaluate the performance of deep learning algorithms across different datasets, i.e., cross-dataset performance evaluation. In machine learning, cross-data dataset performance validations were mostly initiated by the community discussion regarding dataset bias [33], where techniques have been proposed on how to reduce bias during training.

Most of the distracted driver detection algorithms reviewed in Section 3.2 are published with comparative evaluations. For example, Yan *et al.* [13] proposed a CNN-based approach that recognises driving posture based on the position of the hand and evaluated the proposed approach on three datasets. Other authors ([5], [22], [30], [104]) compare the performance of the proposed method with other approaches. However, the focus of these papers is on the proposed algorithms. Recently, Ezzouhri *et al.* [27] evaluated their proposed driver body segmentation-based distracted driver detection algorithm on their custom dataset (EZZ2021) and the AUC2 dataset. The main contribution of the authors was to propose a new algorithm and create a distracted driver detection dataset. However, it is not clear if the authors trained their proposed algorithm on each dataset (EZZ2021 and AUC2 training sets) and tested it on the EZZ2021 and AUC2 test sets.

Recently, Kashevnik *et al.* [105] presented an extensive literature survey on distracted driver detection and outlined the entire chain of distracted driver detection from sensor data acquisition to data preprocessing, behaviour inference, and distraction type inference. Similarly, Huang *et al.* [106] provided an extensive literature survey on vision-based distracted driver detection algorithms. Although these studies are comprehensive and provide current state-of-the-art knowledge on distracted driver detection, none of them evaluates and analyse the performance of distracted driver detection algorithms. Li *et al.* [106] presented a review of distracted driver detection algorithms and then proceeded to evaluate the performance of ten deep learning-based algorithms using the AUC2 dataset. However, cross-dataset evaluations were not conducted.

3.4 Summary

A literature review on deep learning-based distracted driver detection was presented in this chapter in support of the first research objective (ROJ1) of this thesis. The literature review also introduced common datasets used to train and benchmark distracted driver detection algorithms. Convolutional neural networks dominate deep learning-based distracted driver detection due to their ability to extract and learn image features. Two major themes emerge from the literature: CNN feature classification methods and robust distracted driver detection methods. CNN methods extract spatial features from images and use a multilayer perception or a fully connected layer to classify these features.

While CNN feature classification methods have shown impressive results, they often struggle to adapt to new data. Researchers have proposed various approaches, including ensemble methods, driver segmentation methods, and driver ROI detection methods, to address this issue. However,

the generalisation ability of these methods remains a topic of exploration. Few to none of the existing studies have evaluated the cross-dataset performance of deep learning-based distracted driver detection methods. Furthermore, none of the current studies have attempted to use the activities of key body parts involved in driving to improve distracted driver detection.

Overall, the observations made from the literature review suggest that none of the existing methods address the problems identified in Section 1.1.2. As a result, the remaining chapters of this thesis attempt to address the identified research problems by providing more details on the overall research methodology outlined in Section 1.5 (Chapter 4 and Chapter 5) and conducting experiments to evaluate the proposed method (Chapter 6).

Chapter 4

Cross-dataset performance evaluation

This chapter sets out to achieve the second objective (ROJ2) of this thesis. It evaluates how well deep learning-based distracted driver detection algorithms perform on unfamiliar image datasets not part of their training. Specifically, it addresses one critical question: to what extent can deep learning distracted driver detection algorithms generalise on unknown image datasets that were not used for training? This question is addressed by evaluating the performance of state-of-the-art deep learning-based algorithms on widely used benchmark datasets.

Section 4.1 will provide information about the experimental setup used to produce the results presented in Section 4.2. Specifically, the experimental setup will include an overview of the algorithms that were selected for evaluation (Section 4.1.1), the image datasets used (Section 4.1.2), the evaluation metrics that were chosen (Section 4.1.3), as well as the evaluation method used (Section 4.1.4). Additionally, the training procedures and parameters used to train the selected algorithms are also described in Section 4.1.5. Finally, Section 4.3 will summarise the outcomes of the experiment.

The work described in this chapter forms part of the paper published in the Proceedings of the 2022 RAPDASA-RobMech-PRASA-CoSAAMI conference [38].

4.1 Experimental setup

The aim of this experiment is to evaluate the cross-dataset performance of deep learning-based distracted driver detection algorithms. To achieve this aim, METHOD2 introduced in Section 1.5 will be used. The following sections will provide more details about the implementation of METHOD2.

4.1.1 Algorithms

In this experiment, six state-of-the-art algorithms with publicly available code or where authors provided the code upon request are evaluated. In instances where code is not available, similar

algorithms are implemented based on their original publications. The selection of the algorithms was informed by their reported performance in previous research [107], [108]. Further, representative algorithms that are commonly used and recent have been selected.

For evaluation, four primary groups of distracted driver detection algorithms are selected: CNN feature classification with transfer learning, driver segmentation methods, hybrid CNN-RNN methods, and pose estimation-based methods. Table 4.1 presents a comprehensive list of the evaluated algorithms alongside their respective approaches.

Table 4.1: List of algorithms evaluated.

Algorithm	**Approach**
ResNet50 [109]	**CNN feature classification:** Pre-trained ResNet50 model trained using transfer learning.
EffecientNetB0 [101]	**CNN feature classification:** Pre-trained EfficientNetB0 model trained using transfer learning.
Leekha_GrabCut [28]	**Driver segmentation:** CNN feature extraction algorithm trained on images that were preprocessed using an instant segmentation algorithm to remove background noise.
ConvLSTM [110]	**CNN-RNN:** Integrates CNNs and Long Short-Term Memory (LSTM) networks to analyze video sequences or image frames, capturing both spatial and temporal features.
CNN-LSTM [22]	**CNN-RNN:** Combination of Convolutional and LSTM layers.
CNN-Pose [25]	**Pose estimation:** Combines a CNN predictions and predictions of a random forest algorithm trained on detected human key points.

4.1.2 Datasets

The three public image datasets introduced in Section 3.1 are used to evaluate the selected algorithms. The image datasets include the EZZ2021 dataset, the STF dataset, and the AUC2 dataset. The choice of the AUC2 and STF datasets stems from their widespread use as benchmarks for evaluating distracted driver detection algorithms. The EZZ2021 dataset, a more recent addition, shares similarities with the AUC2 and STF datasets. These image datasets are characterised by their relatively large size and the inclusion of nine distinct distracted driving activities, and an additional class for safe driving.

4.1.3 Evaluation metrics

The cross-dataset performance of the selected algorithms is evaluated using two quantitative metrics: accuracy and F1-score [111]. Accuracy, a widely employed and straightforward measure, quantifies the proportion of correct predictions made by a model out of the total number of observations in the test set. The F1-score, also recognized as the F-measure emerges as a pivotal metric.

It represents the weighted harmonic mean of precision and recall performance metrics, expressed as

$$F1 - score = \frac{2 \times (Precision \times Recall)}{Precision + Recall}, \tag{4.1}$$

where recall measures the ability of a model to identify all relevant instances and aims to reduce the error of classifying a positive instance as negative. On the other hand, precision focuses on the accuracy of positive predictions and aims to reduce the error of classifying a negative instance as positive. Precision and recall metrics can be calculated using

$$Precision = \frac{TP}{TP + FP}, and \tag{4.2}$$

$$Recall = \frac{TP}{TP + FN}. \tag{4.3}$$

The variables TP, FP, and FN can be defined as follows [83]:

- **True positive (TP):** Occurs when a model correctly predicts a positive outcome when the actual outcome is indeed positive.
- **False positive (FP):** Occurs when a model incorrectly predicts a positive outcome when the actual outcome is negative.
- **False negative (FN):** Occurs when a model incorrectly predicts a negative outcome when the actual outcome is positive.

The introduction of the F1-score as a metric is motivated by its ability to provide a single, comprehensive measure for evaluating the overall performance of a classification model. The F1-score essentially strikes a balance between recall and precision, synthesising both aspects into a single value. The F1-score ranges from zero to one, with a score of one indicating perfect precision and recall and a score of zero indicating poor performance. Consequently, it provides a more detailed evaluation of model performance.

For more in-depth analysis, the study uses Class Activation Maps (CAMs), which are instrumental in gaining insights into what a Convolutional Neural Network (CNN) "sees" and the reasoning behind its ultimate prediction. Specifically, the study employs an approach known as Grad-CAM [112]. Grad-CAM works by finding the final convolutional layer in the network and then examining the gradient information flowing into that layer. The outcome of Grad-CAM is as a heatmap visualization corresponding to a chosen class label, which can be either the top predicted label or an arbitrary label selected for debugging purposes. This heatmap provides a visual means of confirming the areas in an image that capture the CNN's attention.

4.1.4 Evaluation method

The selected algorithms will be evaluated using the three-step quantitative approach introduced in Section 1.5 (METHOD2). The approach entails:

- **Step 1:** Preparing image datasets.
- **Step 2:** Fine-tuning and training the selected algorithms.

- **Step 3:** Evaluating each method across the test sets prepared in Step 1.

Figure 4.1 illustrates the cross-dataset performance evaluation procedure that is used. In step 1, each distracted driver detection image dataset is divided into three sets: training, validation, and testing. The predefined splits officially released with the image dataset are utilized whenever available. In cases where such predefined splits are not provided, the image dataset is partitioned according to the following ratios: 80% for training data, 10% for validation data, and another 10% designated for the testing set. In the second step, the training sets are exclusively used for model training, while the validation sets serve the purpose of hyperparameter tuning. The test sets are used for cross-dataset performance evaluation in the third and final step. It is important to note that each algorithm is trained independently on each dataset and tested against all three. The training details of each algorithm will be provided in Section 4.1.5.

The methodology employed here focuses on calculating the average cross-dataset performance accuracy, excluding the intra-dataset accuracy in each case. This approach is taken because the model is expected to perform well on a test set from the same dataset as the training set (as explained in Section 3.2.1). Including intra-dataset accuracy would inflate the overall average and not accurately reflect the model's ability to generalise to unseen data from entirely different datasets. This exclusion allows for a more rigorous evaluation of the model's ability to perform well on entirely new data, which is a crucial aspect of real-world application.

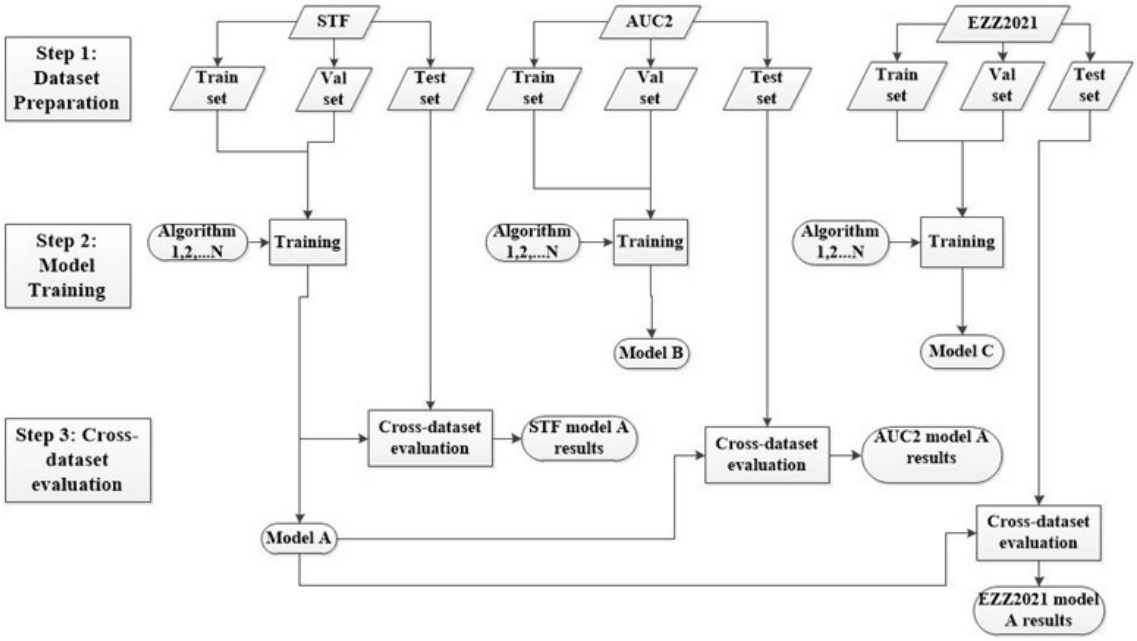


Figure 4.1: Cross-dataset evaluation method employed. Example showing how model **A**, trained on the STF dataset, is evaluated across the three test sets to produce three independent results (STF model A results, AUC2 model A results, and EZZ2021 model A results).

4.1.5 Training procedure

Table 4.2 summarizes the training procedures that are used for each algorithm in the study. The CNN feature classification methods (ResNet50 and EfficientNetB0) were trained using the transfer learning framework introduced in Section 3.2.1. The Leekha_GrabCut approach incorporated the GrabCut background removal algorithm as a preprocessing step with an EfficientNetB0 base. The ConvLSTM model featured 4 ConvLSTM2D recurrent layers, while the CNN LSTM model utilized an AlexNet architecture and an LSTM layer. In the case of CNN-Pose, a combination of fine-tuned EfficientNetB0 and Random Forest models was employed, with coefficients determined using grid search. The algorithms were implemented using Python 3.6 and scikit-learn, NumPy and PyTorch libraries (v1.7.1). All algorithms were trained on a Linux computer with an Nvidia GeForce GTX 1080 graphical processing unit (GPU). The training and validation performance metrics of the algorithms are presented in Section 4.2.1.

Table 4.2: Procedure followed to train the selected algorithms.

Algorithm	**Training procedure**
ResNet50 and EfficientNetB0	ResNet50 and EfficientNetB0 architectures pre-trained on ImageNet were fine-tuned to each of the three datasets using the transfer learning framework. The top layers (head) were replaced by a GlobalAveragePooling2D layer, followed by a Dropout layer and a fully connected layer with ten neurons.
Leekha_GrabCut	For the Leekha_GrabCut algorithm, an EfficientNetB0 model pre-trained on ImageNet was fine-tuned to the three image datasets. The GrabCut background removal algorithm was incorporated as a pre-processing stage to the data pipeline used for training the Leekha_GrabCut algorithm.
ConvLSTM	A convLSTM model with 4 ConvLSTM2D recurrent layers was used. A Maxpooling3D layer and a dropout layer followed each ConvLSTM2D recurrent layer. The Maxpooling3D layer reduces the dimensions of the frames and avoids unnecessary computations. Dropout layers help prevent overfitting the model on the data.
CNN-LSTM	The CNN LSTM model was built using the AlexNet architecture and an LSTM layer with 50 units. A fully connected layer with ten neurons and a softmax activation function was used for class prediction. For both convLSTM and CNN LSTM models, the datasets were prepared as sequence data with five images.
CNN-Pose	The CNN-Pose algorithm consists of a fine-tuned EfficientNetB0 architecture using transfer learning and a Random Forest machine learning model trained on detected human key points obtained through pose estimation. The final prediction was a combination of predictions from the CNN and Random Forest models multiplied by two different coefficients that add up to one. For this study, the coefficients were obtained using a grid search for each dataset.

4.2 Results and analysis

This section presents the training performance metrics of the selected algorithms and the cross-dataset performance evaluation results.

4.2.1 Training results

The algorithms were trained using the hyperparameters shown in Table 4.3. The hyperparameters were obtained through experiments and using the GridSearch technique. Table 4.4 shows the coefficients that were obtained for the CNN-Pose algorithm across the three datasets. To prevent overfitting, early stopping was used to stop the learning process if the validation loss stopped decreasing for ten consecutive training epochs. In addition, the dropout layer technique introduced in Section 2.3.6 was also used.

It is noteworthy that all algorithms were trained using the Adam optimization algorithm [113], which was chosen based on hyperparameter-tuning experiments. The hyperparameters of each algorithm were also kept the same across the three training sets to allow for a fair comparison.

Table 4.3: Hyperparameters used for training the algorithms.

Algorithm	**Learning rate**	**Epochs**	**Optimizer**	**Dropout**
ResNet50	Head: 0.001 Fine-tuning: 1e-5	Head: 15 Fine-tuning: 70	Adam	0.2
EfficientNetB0	Head: 0.001 Fine-tuning: 1e-5	Head: 15 Fine-tuning: 70	Adam	0.2
Leekha_GrabCut	Head: 0.001 Fine-tuning: 1e-5	Head: 15 Fine-tuning: 70	Adam	0.2
ConvLSTM	0.001	70	Adam	0.2
CNN LSTM	0.001	70	Adam	0.25
CNN-Pose	CNN: 0.001	70	Adam	0.2

Table 4.4: Coefficients obtained for the CNN-Pose estimation algorithm.

Dataset	**Coefficients (CNN, Pose)**
EZZ2021	(0.3, 0.7)
STF	(0.3, 0.7)
AUC2	(0.4, 0.6)

Table 4.5 shows the training and validation accuracy performance results of the algorithms. From the results, the difference between the training and validation accuracy of all algorithms is not significant except for one. The Leekha_GrabCut algorithm trained on the AUC2 dataset had 97.78%

training accuracy and 44.92% validation accuracy, which is a significant difference. This suggests that the algorithm may be overfitting the data, or the distribution of the AUC2 training set may differ from that of the AUC2 validation set, which may affect the performance of the Leekha_GrabCut algorithm trained on the AUC2 dataset. However, this is not the case with the other algorithms since the differences between their training and validation accuracy are insignificant. It is important to note that the focus of this thesis is not to optimize the performance of each algorithm on individual datasets.

Table 4.5: Training (Train) and validation (vali) accuracy performance of the algorithms.

Algorithm	**EZZ2021**		**STF**		**AUC2**	
	Train	Vali	Train	Vali	Train	Vali
ResNet50	100	90	99.75	99.70	99.42	97.30
EfficientNetB0	99.99	99.80	99.91	99.80	99.64	97.8
convLSTM	100	100	99.08	99	97.52	92
CNN LSTM	91.90	92.50	92.65	93	80.06	70
Leekha_GrabCut	99.90	99.27	96.47	90.10	97.78	44.92
CNN-Pose estimation	-	95.83	-	93.64	-	94

4.2.2 Cross-dataset performance results

Tables 4.6 through 4.8 present the cross-dataset performance results, measured in terms of accuracy, of the evaluated algorithms. Each table presents the results of the algorithms when trained on one dataset and tested on the three test sets. Based on the results, the following observations can be made:

- The algorithms trained on the EZZ2021 dataset demonstrated impressive performance when evaluated on their corresponding test set, achieving an average accuracy rate of 81.85%. However, it is important to highlight that both the convLSTM and CNN LSTM algorithms yielded accuracies below 70%. In contrast, these algorithms, when trained on the same dataset, demonstrated suboptimal performance when tested on the AUC2 and STF test sets. In summary, the CNN-Pose algorithm emerged as the most robust performer, achieving an average accuracy of 65.6%.
- The algorithms trained on the AUC2 dataset underperformed on the AUC2 test set, as well as the EZZ2021 and STF test sets, achieving average accuracies of less than 40%.
- In a manner consistent with the observations above, the algorithms trained on the STF training dataset performed well on the corresponding STF test set but did not perform well on the AUC2 and EZZ2021 test sets. However, the CNN-Pose algorithm did better on the EZZ2021 test set than on the STF test set.
- Out of the six algorithms evaluated, the CNN-Pose algorithm performed the best on all test sets, regardless of the training dataset used. Following closely, the Leekha_GrabCut algorithm ranks as the second-best performer across all three datasets.

- Even though the CNN-Pose algorithm had a better overall cross-dataset accuracy performance, there is still a significant difference between its intra-dataset accuracy and the average cross-dataset accuracy. For example, when trained on the EZZ2021 dataset, CNN-Pose achieved an intra-dataset of 85.97% and an average cross-dataset accuracy of 65.6%.

Table 4.6: Accuracy performance results of the algorithms when trained on the EZZ2021 training set.*The ***Average*** was calculated using only the STF and AUC2 test sets.

Algorithm	**EZZ2021 test**	**AUC2 test**	**STF test**	***Average****
ResNet50 – EZZ2021	96.18	27.93	31.15	***29.54***
EfficientNetB0 – EZZ2021	87.98	13.87	17.98	***15.93***
convLSTM – EZZ2021	65.03	55.52	8.76	***32.14***
CNN LSTM – EZZ2021	58.19	50.43	8.09	***29.26***
Leekha_GrabCut – EZZ2021	97.74	33.79	30.45	***32.12***
CNN-Pose – EZZ2021	85.97	52.45	78.75	***65.6***
Average	***81.85***	***39***	***29.20***	

Table 4.7: Accuracy performance results of the algorithms when trained on the STF training set.*The ***Average*** was calculated using only the EZZ2021 and AUC2 test sets.

Algorithm	**EZZ2021 test**	**AUC2 test**	**STF test**	***Average****
ResNet50 – STF	16.81	36.28	99.64	***26.55***
EfficientNetB0 – STF	25.19	27.37	99.47	***26.28***
convLSTM – STF	7.85	18.48	99.10	***13.17***
CNN LSTM – STF	10.42	29.86	93.71	***20.14***
Leekha_GrabCut – STF	44.05	33.40	88.30	***38.73***
CNN-Pose – STF	79.92	43.80	73.91	***61.86***
Average	***30.71***	***31.53***	***92.36***	

Table 4.8: Accuracy performance results of the algorithms when trained on the AUC2 training set.*The ***Average*** was calculated using only the EZZ2021 and STF test sets.

Algorithm	**EZZ2021 test**	**AUC2 test**	**STF test**	***Average****
ResNet50 – AUC2	16.27	40.97	34.64	***25.46***
EfficientNetB0 – AUC2	26.62	34.64	43.12	***34.87***
convLSTM – AUC2	20.35	19.94	20.22	***20.29***
CNN LSTM – AUC2	18.75	22.02	22.92	***20.84***
Leekha_GrabCut – AUC2	40.03	44.92	43.84	***41.94***
CNN-Pose – AUC2	48.28	53.79	56.21	***52.25***
Average	***28.38***	***36.05***	***36.83***	

For further analysis, the F1-score was used to compare the performance of the algorithms in the safe driving class. Tables 4.9 through 4.14 show the results of the algorithms when trained and tested on each of the three datasets. The detailed per-class and overall performance of the

algorithms can be found in Appendix B.

From the F1-score analysis, it becomes evident that all algorithms exhibit commendable performance in detecting the "safe driving" class when tested on a test set originating from the same dataset used for training. However, a notable exception arises when algorithms are trained and tested on the AUC2 dataset, where the average F1-scores range from 0.09 to 0.63, indicating a challenge in detecting this class.
Conversely, when these algorithms are confronted with entirely new, previously unseen test datasets, they collectively display suboptimal results. It is noteworthy, however, that both the Leekha_GrabCut and CNN-Pose algorithms consistently demonstrate robust performance across all datasets, with the average cross-dataset F1-score of the CNN-Pose algorithm ranging from 0.61 to 0.79. In stark contrast, the convLSTM and CNN LSTM models exhibit the weakest performance when models are trained on one dataset and tested on completely new test sets. For both algorithms, the average F1-score across all datasets ranges from 0.00 to 0.28, corroborating the earlier findings based on cross-dataset accuracy results.

Table 4.9: F1-score performance results of the ResNet50 model on the safe driving class. *In each case, the ***Average F1-score*** excludes the intra-dataset F1-score.

Safe driving class					
ResNet50		**Test set**			
		EZZ2021	STF	AUC2	***Average F1-score***
Train set	EZZ2021	0.85	0.31	0.52	***0.42***
	STF	0.0	0.95	0.50	***0.25***
	AUC2	0.19	0.39	0.61	***0.29***
	Average F1score	***0.34***	***0.48***	***0.54***	

Table 4.10: F1-score performance results of the EfficientNetB0 model on the safe driving class. *In each case, the ***Average F1-score*** excludes the intra-dataset F1-score.

Safe driving class					
EfficientNetB0		**Test set**			
		EZZ2021	STF	AUC2	***Average F1-score***
Train set	EZZ2021	0.85	0.19	0.27	***0.23***
	STF	0.44	0.88	0.24	***0.34***
	AUC2	0.54	0.39	0.35	***0.47***
	Average F1score	***0.61***	***0.49***	***0.29***	

Table 4.11: F1-score performance results of the convLSTM model on the safe driving class. *In each case, the ***Average F1-score*** excludes the intra-dataset F1-score.

Safe driving class					
convLSTM		**Test set**			
		EZZ2021	STF	AUC2	***Average F1-score***
Train set	EZZ2021	0.97	0.0	0.0	***0.00***
	STF	0.0	0.97	0.23	***0.12***
	AUC2	0.45	0.11	0.35	***0.28***
	Average F1score	***0.47***	***0.36***	***0.09***	

Table 4.12: F1-score performance results of the CNN LSTM model on the safe driving class. *In each case, the ***Average F1-score*** excludes the intra-dataset F1-score.

Safe driving class					
CNN LSTM		**Test set**			
		EZZ2021	STF	AUC2	***Average F1-score***
Train set	EZZ2021	0.62	0.0	0.04	***0.02***
	STF	0.15	0.87	0.09	***0.12***
	AUC2	0.13	0.22	0.39	***0.18***
	Average F1score	***0.30***	***0.36***	***0.17***	

Table 4.13: F1-score performance results of the Leekha_GrabCut model on the safe driving class. *In each case, the ***Average F1-score*** excludes the intra-dataset F1-score.

Safe driving class					
Leekha_GrabCut		**Test set**			
		EZZ2021	STF	AUC2	***Average F1-score***
Train set	EZZ2021	0.94	0.27	0.46	***0.37***
	STF	0.29	0.88	0.27	***0.28***
	AUC2	0.39	0.43	0.58	***0.41***
	Average F1score	***0.54***	***0.53***	***0.44***	

Table 4.14: F1-score performance results of the CNN-Pose model on the safe driving class. *In each case, the ***Average F1-score*** excludes the intra-dataset F1-score.

Safe driving class					
CNN-Pose		**Test set**			
		EZZ2021	STF	AUC2	***Average F1-score***
Train set	EZZ2021	0.99	0.53	0.69	***0.61***
	STF	0.98	0.96	0.60	***0.79***
	AUC2	0.78	0.56	0.60	***0.67***
	Average F1score	***0.92***	***0.68***	***0.63***	

To gain insights into the specific features utilized by the CNN models during the prediction process, the Grad-CAM algorithm was employed on the ResNet50 model. Illustrated in Figure 4.2 and Figure 4.3 are two representative outputs generated by the Grad-CAM algorithm when applied to a ResNet50 model that was trained and tested on the STF dataset, specifically focusing on the "safe driving" and "Make-up" classes. Additional Grad-CAM outputs for the remaining classes are in Appendix C for reference.

Based on the Grad-CAM analysis, the following observations are made:

- It is apparent that the model effectively identifies the relevant features or image regions when making predictions on test images sourced from the same dataset as the training set. In contrast, when confronted with test images from a different dataset, the model appears to experience confusion and directs its focus toward incorrect features.
- It appears that the model is searching for the position of the driver's hands, specifically their forearms in relation to the steering wheel when predicting a "safe driving" posture, as shown in Figure 4.2. However, the model struggles when it only detects one forearm, which is particularly problematic when images are taken up close to the driver. It is hard to distinguish both arms. Therefore, the model encounters difficulties when dealing with scenarios where the driver is positioned close to the camera.
- While the models successfully learn significant features, they also acquire non-essential features, a phenomenon particularly pronounced when the entire image is employed for training. For instance, in the case of the "make-up" class, the models tend to focus on hands near the head, facial features, or the presence of an open front mirror, as shown in Figure 4.3. Notably, an open front mirror can lead to model confusion when analysing other images where the front mirror is open, potentially causing the model to take suboptimal shortcuts.
- The presence of a cell phone around the driver confuses the model in predicting classes that involve the driver using a cell phone.

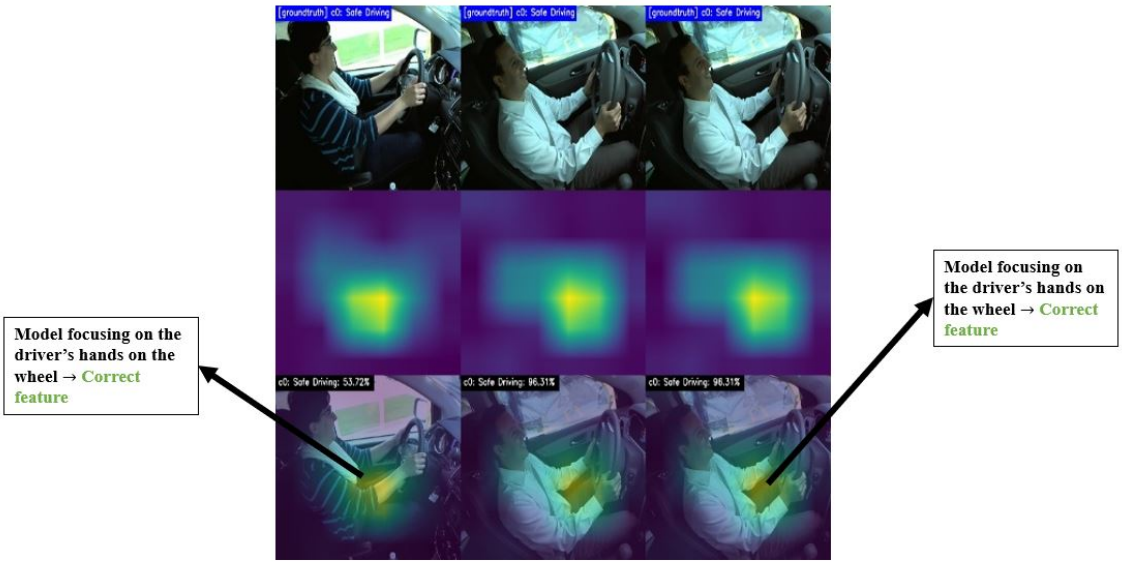


Figure 4.2: Grad-CAM example for the safe driving class.

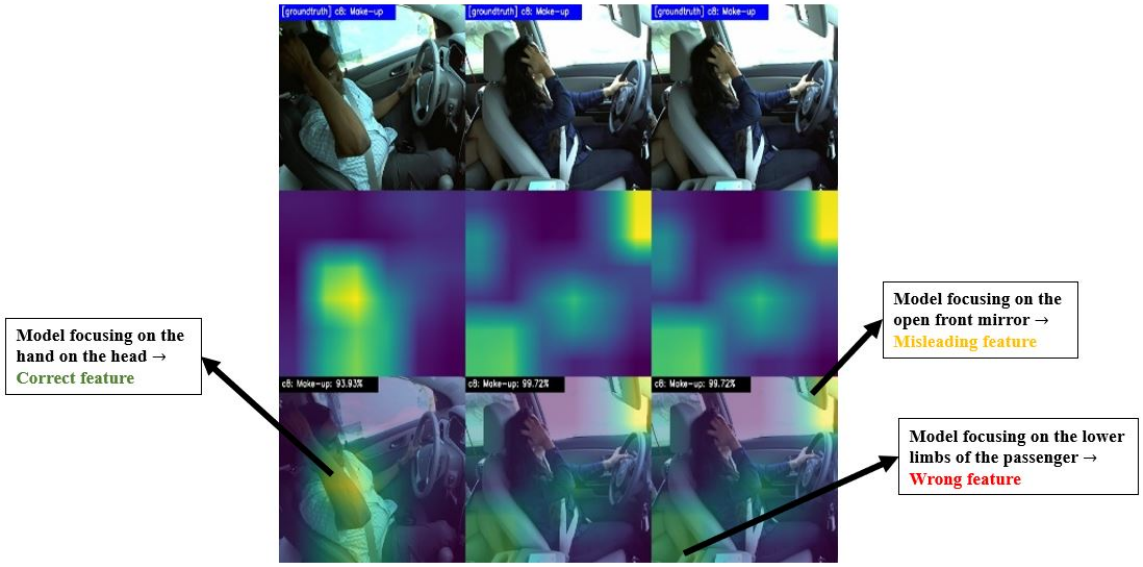


Figure 4.3: Grad-CAM example for the Make-up class.

The results and analysis above suggest the following:

- The CNN-Pose model exhibits superior generalization capabilities compared to the other algorithms, primarily owing to its consistent cross-dataset performance. The robust performance can be attributed to the model's capacity to leverage the rich features learned by the CNN and human key points, which are less variable.
- The Leekha_GrabCut model emerged as the second-best performer among the algorithms evaluated. This notable performance can be attributed to the incorporation of the GrabCut algorithm, which effectively eliminates background noise and directs the model's attention towards the driver's body posture during training. This focus on posture features helps account for the Leekha_GrabCut algorithm's ability to achieve commendable cross-dataset performance as it mitigates dataset-to-dataset variability by eliminating non-essential objects and distractions, thereby enhancing the model's distracted driver detection capabilities.
- The performance of algorithms trained on the AUC2 dataset is adversely affected by certain dataset characteristics. Notably, the primary difference among the three datasets, as indicated by the initial dataset splits, is that in the AUC2 dataset, drivers present in the training set do not appear in the testing set. In contrast, the EZZ2021 and STF datasets feature drivers in the training and testing datasets. Another distinguishing factor is that within the AUC2 dataset, drivers do not partake in all posture activities. These discrepancies offer insights into the challenges faced by models trained on the AUC2 training dataset, which exhibit suboptimal performance on the AUC2 test dataset. Furthermore, this could explain why models trained on the EZZ2021 and STF training datasets perform commendably on their respective testing datasets.

- The poor performance of the algorithms trained on the AUC2 dataset can also be attributed to the fact that the dataset is relatively large but not diverse. Each driver in the dataset performs the same driver activity more than 20 times with very little difference between the image frames. This creates an opportunity for shortcut learning that can easily arise due to a systematic relationship between the driver and the background or context [114].
- CNN models that use the whole image without background noise removal or without considering other features that are less variable do not generalise well to new data. This can be attributed to the fact that the three datasets are relatively large but not diverse.

It has been observed that the performance of deep learning distracted driver detection algorithms is not satisfactory when tested on image datasets that were not used in their original training. This problem is particularly prevalent in CNN models that use the entire image without removing background noise or utilizing less variable features. The reason for this suboptimal performance is that the training datasets used are large but not diverse enough. As a result, the deep learning algorithms resort to shortcut learning, which significantly reduces their ability to generalize to new data.

4.3 Summary

This chapter sought to find the extent to which deep learning distracted driver detection algorithms can generalise to new data that was not used for training. A cross-dataset performance evaluation study was carried out. Based on the results it was found that, in general, deep learning distracted driver detection algorithms do not perform very well on testing datasets that do not come from the same dataset as the training dataset. This observation holds particularly true for CNN models that employ the entire image without the removal of background noise or the utilization of less variable features. In addition, it was found that although current distracted driver detection image datasets are large, they lack diversity – a characteristic that negatively affects their ability to generalise on data. In fact, models often resort to shortcut learning when trained on the current datasets.

Chapter 5

Enhancing cross-dataset performance

The study in Chapter 4 has shown that current deep learning-based methods for detecting distracted drivers struggle to generalise to new, unseen data. This challenge is particularly pronounced in the case of CNN-based methods that utilize the entire image for both training and prediction. The associated issue of limited generalisation ability hinders the seamless application of these methods in real-world scenarios.

This chapter focuses on achieving the third research objective (ROJ3) of this thesis. Specifically, the chapter focuses on developing a robust distracted driver detection approach based on recognising distinctive activities of human body parts involved when a driver is operating a vehicle. To achieve ROJ3, METHOD3 introduced in Section 1.5 will be used. The remaining sections of this chapter will provide more details on the proposed approach (Section 5.1) and its implementation (Section 5.2).

5.1 Proposed approach

The literature review presented in Chapter 3 has revealed that the current ROI detection methods [9], [29]–[31] use object detection to detect a driver region of interest and common distraction objects such as cell phones and drinking bottles. The detected ROI is then used to classify the driver behaviour. Figures 5.1 and 5.2 illustrate how current driver ROI detection methods use object detection.

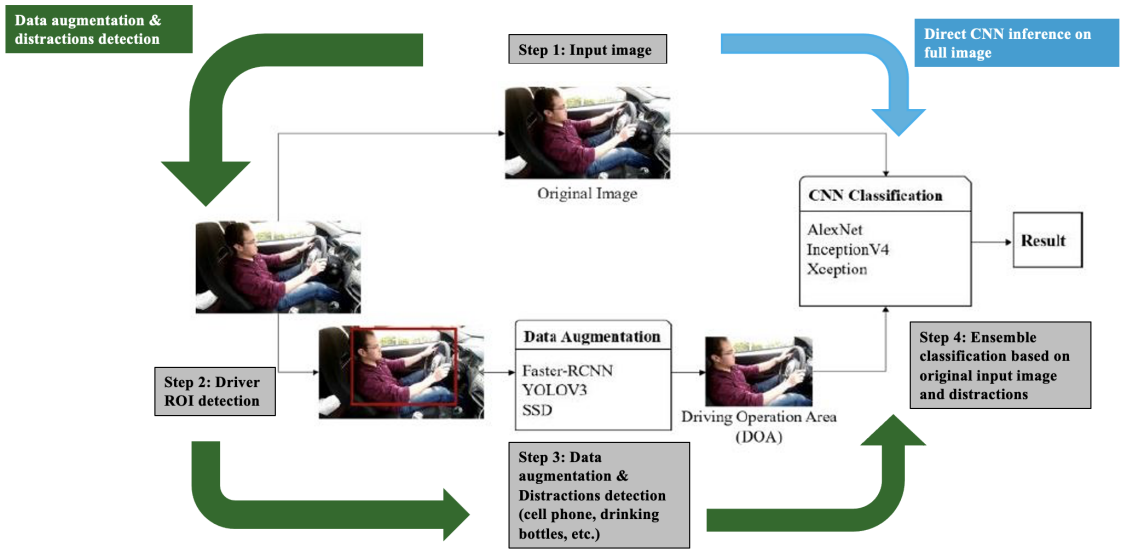


Figure 5.1: Driver ROI detection approach proposed by Wang *et al.* [30].

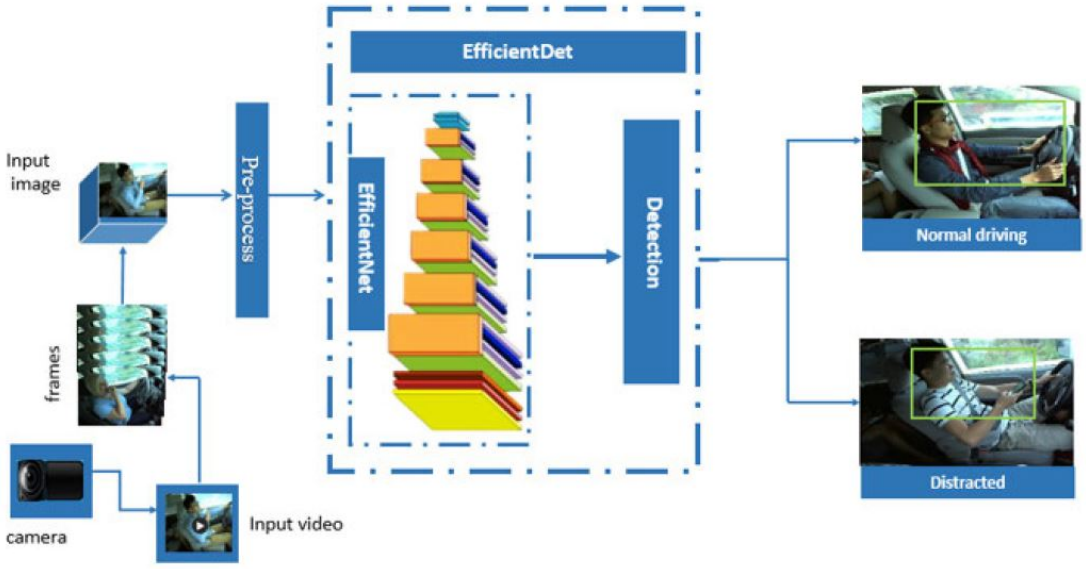


Figure 5.2: Driver ROI detection approach proposed by Sajid *et al.* [29].

Contrary to current driver ROI detection methods, the proposed approach uses object detec-

tion to identify important human body parts when a driver operates a vehicle and classify their states into activities in one forward pass. These activities are used to predict if a driver is distracted.

The proposed approach consists of two key steps. In the first step, the driver's body parts are detected, and the state of each body part is classified into an activity. The second step utilizes the detected activities to make a final prediction using a simple decision tree-based approach. Inspired by the findings of Yang *et al.* [99], the distracted driver classification problem in the second step is treated as a binary classification problem, focusing on determining whether the driver is distracted or not. The specific tasks involved in each step are detailed below. Figure 5.3 provides an overview of the proposed approach.

In step 1, the driver's head and hands are detected. The state of the driver's head is classified as either "eyes on the road" or "eyes off the road." The positions of the left and right hands in relation to the steering wheel are used to classify whether both hands are on the wheel or if one hand is on the wheel. Furthermore, common distracting activities that a driver may engage in while driving are recognized. These activities include cell phone usage, drinking, hands on the face, and talking on the phone. The activities are identified by detecting the presence of a cell phone or any object that can be used for drinking and classifying the activity based on the detected object. In the second step of the approach, the final prediction is made by evaluating two conditions: "eyes on the road" and "both hands on the steering wheel." If both conditions are satisfied, the driver is considered to be in a safe driving position. Conversely, if one of the conditions is not met, the driver is deemed distracted.

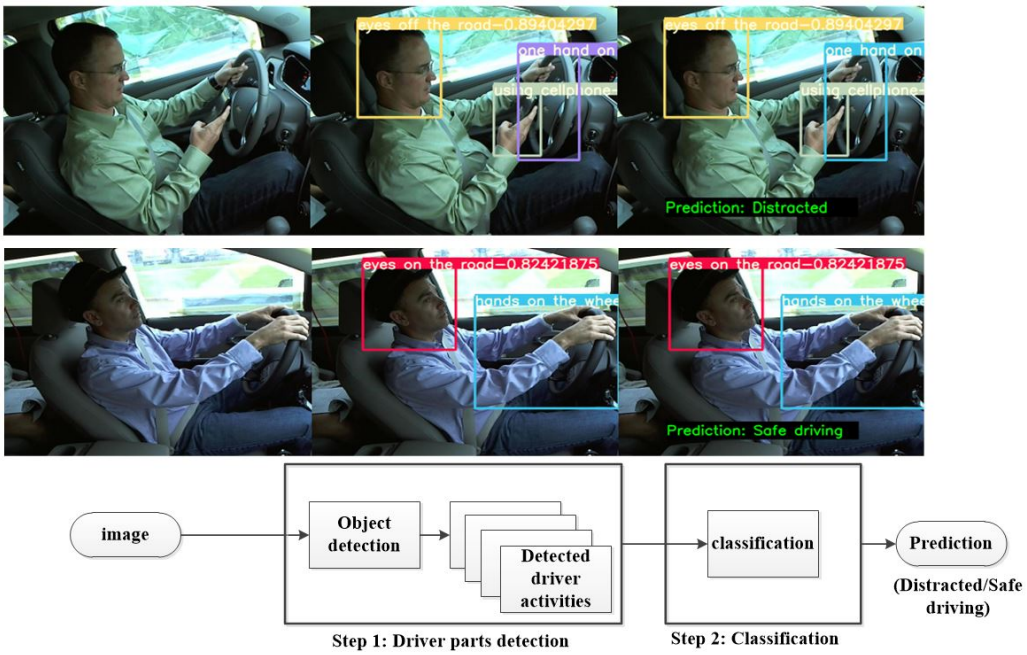


Figure 5.3: Proposed framework for distracted driver detection. **Step 1:** The driver's body parts are detected, and the activity of each body part is classified. **Step 2:** The detected activities are used to make a final prediction using a decision tree-based approach.

5.2 Implementation

The proposed approach was implemented following the process introduced in Section 2.8. For reference purposes, the overall development process is shown again in Figure 5.4. The overall implementation process starts with project scoping. Project scoping involves problem definition, data definition, and selecting an optimising metric and the satisficing metric. For this thesis, the problem was defined in Section 1.1. The nature of the problem dictates that it should be solved using computer vision methods, specifically object detection. In this thesis, the YOLOv7 model was chosen due to its superiority in terms of accuracy and inference speed compared to other object detectors, as explained in Section 2.6. The average mean precision (mAP) will be the optimising parameter. Specifically, the mAP of the model should be at least ≥ 0.5 for the model's performance to be considered "satisfactory" [85]. The inference time of the model could serve as the satisficing metric. However, a threshold value will not be specified since the focus of this thesis is only limited to improving the robustness of distracted driver detection.

The remaining sections provide more details on each step involved in the implementation process.

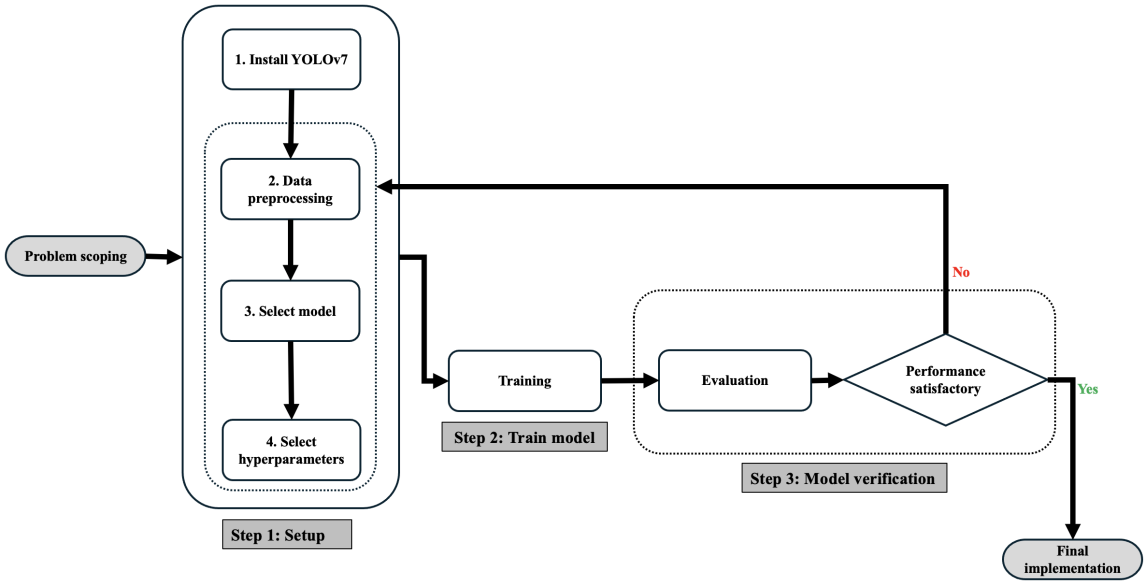


Figure 5.4: Overall development process to train and fine-tune the YOLOv7 model to custom data.

5.2.1 YOLOv7 model setup

The first step of the implementation process (YOLOv7 model setup) involves four key items: installing the YOLOv7 model, preprocessing the data, selecting a model, and selecting hyperparameters. Details pertaining to each item are as follows:

- **Install YOLOv7:** The process of setting up the YOLOv7 model starts with installing the YOLOv7 official code repository. The source code was cloned from the official GitHub repository released with the YOLOv7 paper [36]. After that, the dependencies of the YOLOv7 source code were installed in a conda virtual environment. Figure 5.5 shows the activities associated with the first item of step one of the implementation process.

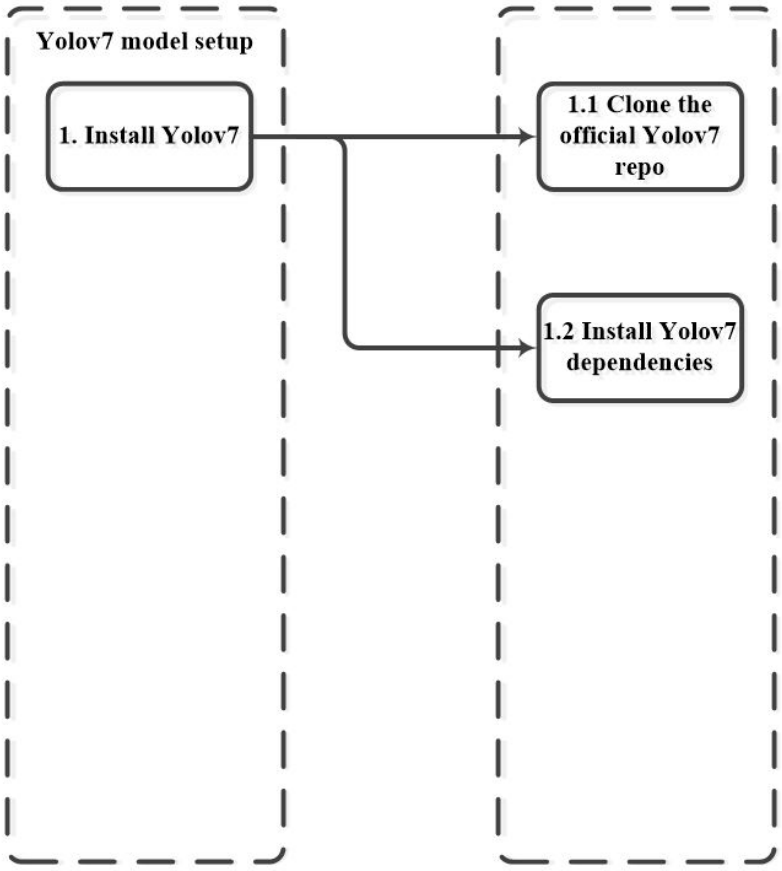


Figure 5.5: Step 1.1: Installation.

- **Data preprocessing:** Figure 5.6 shows the activities associated with data preprocessing. The STF dataset was selected to train the proposed approach. This choice was based on the fact that the STF dataset has many participants (44) compared to the EZZ2021 (9) and AUC2 (26) datasets. As a result, the dataset is more diverse than the EZZ2021 and AUC2 datasets. The diversity of the STF dataset will allow the YOLOv7 model to learn from more different samples during training, thereby improving the generalisation of the model. To align with the proposed approach, which centres on specific regions for prediction, a random selection of 3346 images was allocated for training, while an additional 1,000 images were reserved for validation purposes. The original test split, comprising 2,257 images, was retained for testing. The selected training and validation images were manually annotated with nine labels, which include "eyes on the road", "eyes off the road", "hands on the wheel", "hands on the face", "no hands on the wheel", "one hand on the wheel", "using cell phone", "talking on the phone", and "drinking". The images were annotated using the labelimg[1] tool. The decision to use only a portion (3346/19173) of the STF training set is based on the fact that annotating the

[1]Labelimg: https://github.com/HumanSignal/labelImg

images takes a lot of time. It took a full month and three weeks to annotate the selected 4346 images (training and validation).

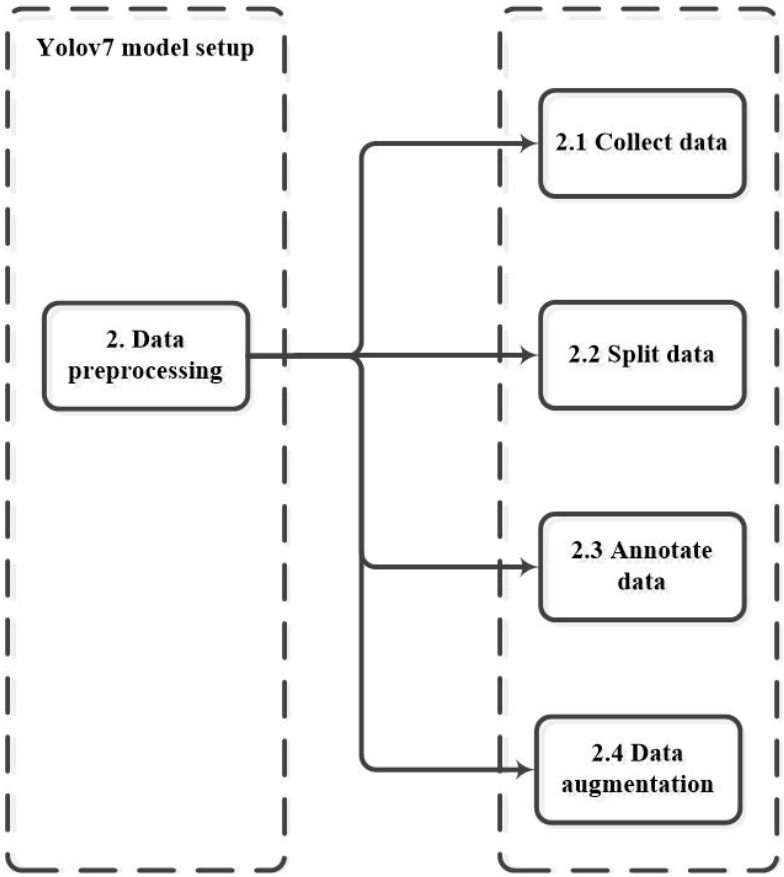


Figure 5.6: Step 1.2: Data preprocessing.

- **Select model:** Figure 5.7 shows the activities associated with model selection. The YOLOv7 model was released with six versions with varying parameters and processing speed measured in frames per second (FPS), as shown in Table **??**. In this work, the standard YOLOv7 model with 36.9 million parameters (YOLOv7) was selected based on a better mAP during experiments.

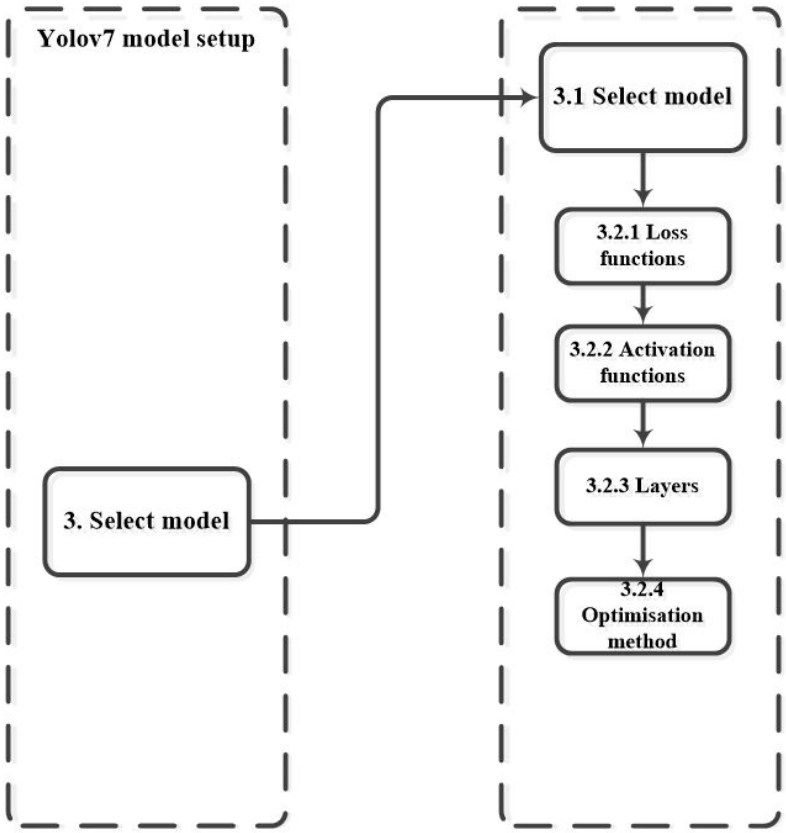


Figure 5.7: Step 1.3: Model selection.

Table 5.1: Official YOLOv7 model versions. * FPS comparisons were done on the Tesla V100 GPU.

Model	**Parameters**	$AP^{\mathbf{test}}$[%]	FPS^*
YOLOv7	36.9M	51.4	161
YOLOv7-X	71.3M	53.1	114
YOLOv7-W6	70.04M	54.9	84
YOLOv7-E6	97.2M	56.0	56
YOLOv7-D6	154.7M	56.6	44
YOLOv7-E6E	151.7M	56.8	36

- **Hyperparameters:** The final item in setting up the YOLOv7 model involves selecting hyperparameters such as the number of epochs, batch size and image size. There are other hyperparameters, such as initial learning rate and box loss gain, as shown in Figure 5.8. This thesis focused on four hyperparameters, which include epochs, image size, IoU_t, and anchor_t. The other hyperparameters were set to their default values. Epochs refer to the number of complete passes through the entire training set. During an epoch, the model sees

and learns from every example in the training dataset once. Image size refers to the input image size during training. IoU_t represents the IoU threshold applied during training. The anchor box threshold value used during training is represented by anchor_t. These four hyperparameters were selected based on their influence on the model's performance (mAP) during hyperparameter-tuning experiments.

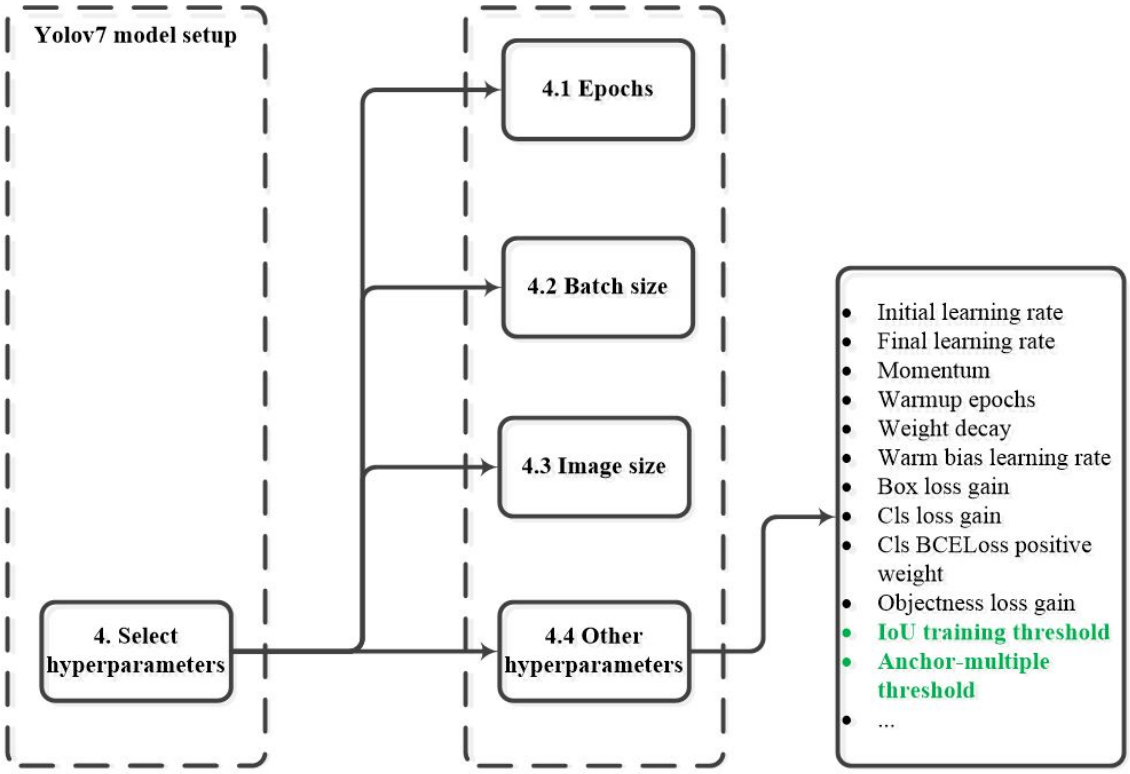


Figure 5.8: Step 1.4: Selecting hyperparameters.

5.2.2 Training

To detect distracted drivers, a standard YOLOv7 model pre-trained on the Microsoft COCO dataset was fine-tuned to the annotated dataset. The training process involved using different hyperparameters for each experiment, which is explained in detail in Chapter 6.

5.2.3 Model evaluation

The focus of this chapter is on developing a robust distracted driver detection method. As a result, the performance of the proposed approach is evaluated using the cross-dataset performance evaluation method presented in Section 4.1.4. Therefore, in addition to the STF test set, the proposed method will be evaluated on the EZZ2021 test set, the AUC2 test set, and the CSIR test

set.

The performance of the proposed approach will be evaluated using the accuracy and F1-score metrics presented in Section 4.1.3.

5.3 Summary

This chapter presented an object detection-based method to improve the generalisation ability of distracted driver detection. In contrast to current driver ROI detection methods, which use object detection to detect a region that consists of the driver, the proposed method uses object detection to detect important human body parts involved when a driver operates a vehicle. The states of detected driver body parts are classified into activities. These activities are used to obtain the final prediction. Treating the problem as a binary classification problem, the final prediction is obtained by evaluating two conditions (activities): "eyes on the road" and "hands on the wheel". If both conditions are satisfied, the driver is considered to be in a safe driving position. Conversely, if one of the conditions is not met, the driver is deemed distracted.

The chapter also provides details on how the proposed method was implemented. Chapter 6 presents experimental results that aim to demonstrate the generalisation ability of the proposed method.

Chapter 6

Experimental results and discusion

Chapter 5 provided details on the proposed approach and its implementation. However, its effectiveness remains unknown. This chapter seeks to bridge this gap by evaluating the performance of the proposed method on the three test sets introduced in Section 3.1.1. In addition, the proposed approach will be evaluated on the CSIR test set introduced in Section 3.1.2. These activities support the fourth research objective of this study (ROJ4).

This chapter consists of two experiments that attempt to answer the following questions:

- **Q1:** Can the cross-dataset performance of convolutional neural network-based distracted driver detection be improved through the detection of driver body parts and classification of their state into activities?
- **Q2:** Can the performance of the proposed approach be improved through the adoption of multiple class-specific object detectors as opposed to a single multi-class object detector?

Section 6.1 provides details about the main experiment that seeks to address Q1. Section 6.2 describes the second experiment to address Q2. The chapter concludes by providing a summary of the outcomes of the two experiments in Section 6.3.

6.1 Main experiment: proposed approach

This experiment aims to evaluate the cross-dataset performance of the proposed approach. The Yolov7 model pre-trained on the Microsoft COCO dataset was fine-tuned on the annotated STF dataset, comprising nine distinct classes representing various driver activities. The nine activities include "eyes on the road," "eyes off the road," "hands on the wheel," "one hand on the wheel," "no hands on the wheel," "hands on the face," "using a cell phone," "talking on the phone," and "drinking". The Yolov7 model was trained for 150 epochs using a 680×680 input image size, an IoU threshold value of 0.50, and an anchor box threshold of 9. These hyperparameters were obtained through a series of hyperparameter-tuning experiments. As mentioned in Section 5.2.1, the other hyperparameters were set to their defaults.

Table 6.1 shows the hyperparameters used to train the Yolov7 model along with the corresponding training performance metrics (mAP@0.5 and mAP@0.95). The trained model obtained an mAP@0.5 and mAP@0.95 of 0.964 and 0.704, respectively. These performance metrics are above the minimum mAP target of 0.5, suggesting that the model was able to learn features from the annotated STF dataset.

Table 6.1: Hyperparameters used to train the Yolov7 model and the corresponding training performance metrics.

Experiment	**epochs**	**img_size**	**iou_t**	**anchor_t**	**mAP@0.5**	**mAP@0.95**
Proposed method	150	680	0.50	9.0	0.964	0.704

6.1.1 Baselines

The performance of the proposed approach was compared to that of the CNN feature classification methods (ResNet50 and EfficientNetB0) and the Leekha_GrabCut algorithm. These algorithms were introduced in Section 4.1.1. The ResNet50 and EfficientNetB0 models, initially pre-trained on the ImageNet dataset, were fine-tuned using the modified STF training dataset through the transfer learning framework. The top layers of these architectures were replaced with a GlobalAveragePooling2D layer, followed by a dropout layer and a fully connected layer with two neurons. A softmax activation function was used on the output layer. Two neurons were used in the output layer since distracted driver detection was treated as a binary classification problem. Table 6.2 presents the hyperparameters used for training. Table 6.2 also shows the accuracy of algorithms on the STF validation set. The ResNet50, EfficientNetB0 and Leekha_GrabCut algorithms acheived validation accuracies of 99.70%, 99.90%, and 99.10%, respectively.

Table 6.2: Hyperparameters used to train the three baseline algorithms.

Algorithm	**Learning rate**	**Epochs**	**Optimizer**	**Dropout**	$Accuracy^{val}$
ResNet50	Head: 0.001	15	Adam	0.2	
	Fine-tuning: 1e-5	70	Adam	-	99.70%
EfficientNetB0	Head: 0.001	15	Adam	0.2	
	Fine-tuning: 1e-5	70	Adam	-	99.80%
Leekha_GrabCut	Head: 0.001	15	Adam	0.2	
	Fine-tuning: 1e-5	70	Adam	-	90.10%

6.1.2 Results and analysis

Table 6.3 shows the performance of the proposed method and the other three CNN-based approaches. From the results, the following observations can be made:

- All approaches perform well on the STF test dataset, with an average accuracy of 95.68%.
- All approaches do not perform well on the CSIR test dataset, with an average accuracy of 56.88%.

- The proposed approach demonstrates a better overall cross-dataset performance than the other three CNN-based methods. Compared to the second-best algorithm (Leekha_GrabCut), the proposed approach improves cross-dataset accuracy performance by 7.8%.
- Out of all the approaches evaluated, the ResNet50 model achieved the lowest overall cross-dataset accuracy, with an average of 71.94%.

Table 6.3: Accuracy performance of the algorithms across the four test sets. *The ***Average*** accuracy was calculated using only the EZZ2021, AUC2 and CSIR test sets.

Algorithm	**EZZ2021**	**STF**	**AUC2**	**CSIR**	***Average****
Proposed method	90.91	96.62	91.62	62.63	*81.72*
ResNet50	82.33	95.91	79.70	53.80	*71.94*
EfficientNetB0	90.94	94.35	82.68	53.18	*75.60*
Leekha_GrabCut	90.73	94.84	82.50	54.21	*75.81*
Average	***94.16***	***96.66***	***79.70***	***56.88***	***76.27***

For further analysis, the F1-score was used to evaluate and compare the balanced performance of the algorithms. Table 6.4 shows the F1-score results of the approaches. Based on the results, the following observations can be made:

- The approaches demonstrate commendable performance on the STF test dataset, achieving an average F1-score of 0.78.
- Conversely, all approaches exhibit poor performance on the CSIR test dataset, with an average F1-score of 0.19.
- The proposed approach achieved the best overall balanced distracted driver detection performance. It is worth noting that the proposed approach achieves the highest accuracy in three out of the four test datasets (STF, AUC2, and CSIR) in Table 6.3, highlighting its effectiveness across various metrics.
- On the AUC2 test set, the ResNet50 model and the EfficientNetB0 model both achieved an F1-score of 0. These zero F1-scores mean that neither model correctly predicted the safe driving class. The ResNet50 model also had low F1-scores on the CSIR and EZZ2021 test sets, measuring 0.02 and 0.05, respectively. As a result, the ResNet50 model performed the worst in detecting distracted drivers, followed by the EfficientNetB0 model.

Table 6.4: F1-scores of the algorithms across the four test sets. *The ***Average*** F1-score was calculated using only the EZZ2021, AUC2 and CSIR test sets.

Algorithm	**EZZ2021**	**STF**	**AUC2**	**CSIR**	***Average***
Proposed method	0.36	0.86	0.75	0.42	*0.51*
ResNet50	0.05	0.79	0.00	0.02	*0.02*
EfficientNetB0	0.29	0.68	0.26	0.00	*0.18*
Leekha_GrabCut	0.21	0.73	0.27	0.10	*0.19*
Average	***0.32***	***0.78***	***0.38***	***0.19***	***0.23***

The results and analysis presented above reveal the following insights:

- The proposed approach demonstrates a significant improvement in the cross-dataset performance of CNN-based distracted driver detection. Compared to the best-performing approach among the three CNN-based methods (Leeka_GrabCut), the proposed approach achieved an overall improvement of 7.8% in classification accuracy performance. The overall balanced performance, as measured by the F1-score, was substantially improved by a factor of 2.68. These improvements are attributed to the fact that the proposed approach focuses on crucial driver body parts and their associated activities, reducing input variance from dataset to dataset.
- All approaches exhibit satisfactory performance on the STF test dataset, which was expected since all algorithms were trained on the STF training dataset. The similarity in characteristics between the STF training and STF test datasets, such as camera viewpoint, drivers used, and cars used, contributes to this favourable performance.
- The algorithms face challenges with the custom CSIR test set due to poor image quality, especially in difficult lighting conditions (see Figure 6.1).
- The poor performance of the algorithms on the CSIR test set indicates that there is still much work to be done to make distracted driver detection algorithms suitable for real-world deployment and integration.

Figure 6.1: Sample qualitative results from the CSIR dataset.

Overall, the findings of this experiment highlight the effectiveness of the proposed approach in enhancing the cross-dataset performance of distracted driver detection while also emphasizing the impact of dataset characteristics on a learning algorithm.

6.2 Multi-class object detection vs multiple class-specific object detectors

The primary objective of this experiment is to evaluate whether the performance of the proposed approach can be improved through the adoption of multiple class-specific object detectors as opposed to a single multi-class object detector. In addition to the proposed approach used in the main experiment in Section 6.1, a second approach that uses class-specific object detectors was

implemented. By contrasting the outcomes derived from these two approaches, it becomes possible to determine the effectiveness of employing class-specific object detectors for enhanced performance.

The class-specific approach entailed the training of three distinct Yolov7 models, each with a specialized focus. The first model concentrated exclusively on detecting the driver's head and discerning whether their eyes were on the road. The second model targeted the driver's hands and the steering wheel to identify two specific driver activities: "both hands on the wheel" and "one hand on the wheel." Finally, the third model was dedicated to detecting four distracting activities that drivers might engage in while driving, namely, "hands on the face," "using a cell phone," "talking on the phone," and "drinking."

All the models were trained using hyperparameters obtained through a series of hyperparameter-tuning experiments. The performance of all the models, measured in terms of mean average precision (mAP) on the validation dataset, is presented in Table 6.5.

Table 6.5: Training details: multi-class object detection vs. multiple class-specific object detectors experiment.

Experiment	**epochs**	**img_size**	**iou_t**	**anchor_t**	**mAP@0.5**	**mAP@0.95**
Yolov7_multi-class	150	680	0.50	9.0	0.964	0.704
Yolov7_class-specific (head)	100	1280	0.50	4.0	0.902	0.7438
Yolov7_class-specific (hands)	100	1280	0.20	4.0	0.988	0.7929
Yolov7_class-specific (distractions)	150	1280	0.20	4.0	0.925	0.7038

6.2.1 Results and analysis

Table 6.6 displays the accuracy performance of two distinct approaches: the Yolov7-Multi-class approach and the Yolov7-Class-specific approach. The results indicate that the Yolov7-Multi-class approach excels on the AUC2 and CSIR test datasets but demonstrates suboptimal performance on the EZZ2021 and STF test datasets compared to the Yolov7-Class-specific approach. In summary, the Yolov7-Multi-class approach maintains a slightly higher average accuracy rate of 81.72% than the Yolov7-Class-specific approach, which achieves an average accuracy of 81.06%.

The balanced performance of the two algorithms was also compared using the F1-score metric, as shown in Table 6.7. Based on the F1-score performance, it can be observed that the Yolov7-Class-specific approach outperforms the Yolov7-Multi-class approach only on the EZZ2021 test dataset, exhibiting a 0.9167 improvement. However, the Yolov7-Class-specific approach demonstrates a superior overall F1-score across all four test datasets.

Table 6.6: Accuracy performance comparison: multi-class object detector vs. class-specific object detectors. *The ***Average*** accuracy was calculated using only the EZZ2021, AUC2 and CSIR test sets.

Algorithm	EZZ2021	STF	AUC2	CSIR	Average*
Yolov7-Multi-class	90.91	96.62	91.62	62.63	81.72
Yolov7-Class-specific	94.16	96.66	88.45	60.57	81.06
Improvement	**3.57%**	**0.04%**	**-3.46%**	**-3.29%**	**-0.81%**

Table 6.7: F1-score performance comparison: multi-class object detector vs. class-specific object detectors. *The ***Average*** F1-score was calculated using only the EZZ2021, AUC2 and CSIR test sets.

Algorithm	EZZ2021	STF	AUC2	CSIR	Average
Yolov7-Multi-class	0.36	0.86	0.75	0.42	0.51
Yolov7-Class-specific	0.69	0.85	0.61	0.40	0.56
Improvement	**0.9167**	**-0.0118**	**-0.1867**	**-0.0476**	**0.098**

Based on the analysis of the results in Tables 6.6 and 6.7, the Yolov7-Class-specific approach does not enhance distracted driver detection accuracy, as it exhibits a marginal decrease of 0.81% in overall accuracy. However, the F1-score results paint a different picture, indicating that the Yolov7-Class-specific approach improves the overall balanced performance by 0.098.

In summary, the findings of this experiment suggest that although using multiple class-specific object detectors leads to a minor reduction in the overall accuracy of distracted driver detection, there is a notable improvement of 0.098 in the overall balanced performance, as indicated by the F1-score. For a more comprehensive understanding of algorithm performance, readers are encouraged to reference the confusion matrices presented in Appendix D of this work. These matrices offer a detailed breakdown of the models' performance, delineating correct and incorrect predictions for each class.

6.3 Summary

This chapter provided experimental results that sought to answer two questions:

- **Main experiment:** Can the cross-dataset performance of convolutional neural network-based distracted driver detection be improved through the detection of driver body parts and classification of their state into activities?
- **Multi-class object detector vs. class-specific object detectors:** Can the performance of the proposed approach be improved through the adoption of multiple class-specific object detectors as opposed to a single multi-class object detector?

The main experiment implemented the proposed approach using the implementation process introduced in Section 5.2. The performance of the proposed approach was compared to three other

CNN-based methods, which served as baselines. Based on the experimental results, the proposed approach improves cross-dataset performance. A cross-dataset accuracy improvement of 7.8% was observed. Most importantly, the overall balanced (F1-score) performance was improved by a factor of 2.68. The experimental results also revealed that although the proposed approach demonstrates commendable performance on the CSIR test set, all algorithms encountered challenges when dealing with the custom CSIR test set, mainly due to lower image quality and difficult lighting conditions.

The second experiment implemented the proposed approach and a variant of the proposed approach that uses multiple class-specific objectors instead of one multi-class object detector. The findings of the second experiment revealed that although using multiple class-specific object detectors leads to a minor reduction in the overall accuracy of distracted driver detection, there is a notable improvement of 0.098 in the overall balanced performance, as indicated by the F1-score.

Chapter 7

Conclusions and future work

7.1 Conclusion

Distracted driving remains a pressing issue globally. Measurement methods for distracted driver detection will remain very prevalent to the early detection of distracted drivers and consequently reducing the number of accidents caused by distracted driving. Methods based on convolutional neural networks have reported remarkable results.

This study has demonstrated that despite the remarkable results reported in the literature, the performance of current deep learning-based distracted driver detection methods is limited in cross-dataset testing scenarios. Representative state-of-the-art deep learning algorithms were trained and tested across different image datasets. The analysis of the results has indicated that one of the major reasons for the lack of consistent performances across different datasets is the lack of large and diverse image datasets. In addition, algorithms that do not use the whole image for training and prediction have a better cross-dataset performance than algorithms that use the whole image.

The proposed approach, which targets important human body parts involved when a driver operates a vehicle, has demonstrated the potential to improve cross-dataset performance significantly. However, further work is required to develop a computer vision-based system that can be deployed in a real-world environment.

7.2 Limitations

In this work, several limitations need to be addressed. First, the major limitation is that the proposed approach was not tested in a real-world environment (in a car). Second, the approach was trained and tested with static images offline. The performance of the approach on video data is not known. Finally, the image datasets used for training and testing were captured in daylight conditions in a well-lit environment.

7.3 Future work

Based on the findings of the study, future work could focus on several aspects:

- The current work has identified the deficiencies of existing distracted driver detection image datasets. Creating large and diverse distracted driver detection image datasets encompassing different driving scenarios and environmental factors could help improve the generalizability and robustness of the proposed approach and deep learning-based algorithms. To reduce the effort required, synthetic image data generation using AI (for example, generative adversarial networks (GANs)) and CGI can be explored.
- The CNN-pose model demonstrated robust dataset-to-dataset performance. In addition, the pose estimation model was given more weight than the CNN in the CNN-Pose algorithm. Future work can focus on designing more robust features using detected human key points.
- Testing the developed models in the real-world would provide valuable insights into their practical effectiveness and potential for integration into real-time driver monitoring systems.
- In the proposed approach, the performance of the Yolov7 model was not compared to the performance of other object detectors such as Yolov8 and Faster R-CNN. Future work can test and compare the performance of the proposed approach with different object detector models.
- Finally, specialised domain generalisation approaches can be explored for deep learning distracted driver detection.

Bibliography

[1] WHO. "Road traffic injuries." Accessed on March 07, 2022. (2022), [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries.

[2] RTMC. "State of road safety report 2019." Accessed on March 07, 2022. (2019), [Online]. Available: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.rtmc.co.za/images/rtmc/docs/traffic_reports/calendar/caldec2019.pdf.

[3] Q. Wu, "An overview of driving distraction measure methods," in *2009 IEEE 10th International Conference on Computer-Aided Industrial Design & Conceptual Design*, IEEE, 2009, pp. 2391–2394.

[4] Y. S. T. Shafeeq Kanaan Shakir AL-Doori and M. Koklu, "Distracted driving detection with machine learning methods by cnn based feature extraction," *International Journal of Applied Mathematics Electronics and Computers*, vol. 9, no. 4, pp. 116–121, 2021.

[5] M. R. Arefin, F. Makhmudkhujaev, O. Chae, and J. Kim, "Aggregating cnn and hog features for real-time distracted driver detection," in *2019 IEEE International Conference on Consumer Electronics (ICCE)*, IEEE, 2019, pp. 1–3.

[6] NHTSA. "Nhtsa releases 2020 traffic crash data." Accessed on March 07, 2022. (2022), [Online]. Available: https://www.nhtsa.gov/press-releases/2020-traffic-crash-data-fatalities.

[7] M. Ngxande, "Correcting inter-sectional accuracy differences in drowsiness detection systems using generative adversarial networks (gans)," Advisors: Prof. Jules-Raymond Tapamo and Dr Michael Burke, Ph.D. thesis, University of Kwa-Zulu Natal, 2020.

[8] P. Papantoniou, E. Papadimitriou, and G. Yannis, "Review of driving performance parameters critical for distracted driving research," *Transportation research procedia*, vol. 25, pp. 1796–1805, 2017.

[9] B.-T. Dong and H.-Y. Lin, "An on-board monitoring system for driving fatigue and distraction detection," in *2021 22nd IEEE International Conference on Industrial Technology (ICIT)*, IEEE, vol. 1, 2021, pp. 850–855.

[10] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, IEEE, 2010, pp. 253–256.

[11] B. News. "Tesla in fatal california crash was on autopilot." Accessed on October 20, 2022. (2018), [Online]. Available: https://www.bbc.com/news/world-us-canada-43604440.

[12] S. Yan, Y. Teng, J. S. Smith, and B. Zhang, "Driver behavior recognition based on deep convolutional neural networks," in *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, IEEE, 2016, pp. 636–641.

[13] C. Yan, F. Coenen, and B. Zhang, "Driving posture recognition by convolutional neural networks," *IET Computer Vision*, vol. 10, no. 2, pp. 103–114, 2016.

[14] F. R. da Silva Oliveira and F. C. Farias, "Comparing transfer learning approaches applied to distracted driver detection," in *2018 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, IEEE, 2018, pp. 1–6.

[15] H. M. Eraqi, Y. Abouelnaga, M. H. Saad, M. N. Moustafa, *et al.*, "Driver distraction identification with an ensemble of convolutional neural networks," *Journal of Advanced Transportation*, vol. 2019, 2019.

[16] C. Huang, X. Wang, J. Cao, S. Wang, and Y. Zhang, "Hcf: A hybrid cnn framework for behavior detection of distracted drivers," *IEEE access*, vol. 8, pp. 109 335–109 349, 2020.

[17] C. Ou and F. Karray, "Enhancing driver distraction recognition using generative adversarial networks," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 3, pp. 385–396, 2019.

[18] J. Cronje and A. P. Engelbrecht, "Training convolutional neural networks with class based data augmentation for detecting distracted drivers," in *Proceedings of the 9th International Conference on Computer and Automation Engineering*, 2017, pp. 126–130.

[19] B. Qin, J. Qian, Y. Xin, B. Liu, and Y. Dong, "Distracted driver detection based on a cnn with decreasing filter size," *IEEE transactions on intelligent transportation systems*, vol. 23, no. 7, pp. 6922–6933, 2021.

[20] M. H. Alkinani, W. Z. Khan, Q. Arshad, and M. Raza, "Hsddd: A hybrid scheme for the detection of distracted driving through fusion of deep learning and handcrafted features," *Sensors*, vol. 22, no. 5, p. 1864, 2022.

[21] C. Streiffer, R. Raghavendra, T. Benson, and M. Srivatsa, "Darnet: A deep learning solution for distracted driving detection," in *Proceedings of the 18th acm/ifip/usenix middleware conference: Industrial track*, 2017, pp. 22–28.

[22] J. M. Mase, P. Chapman, G. P. Figueredo, and M. T. Torres, "A hybrid deep learning approach for driver distraction detection," in *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, IEEE, 2020, pp. 1–6.

[23] F. Nel and M. Ngxande, "Driver activity recognition through deep learning," in *2021 Southern African Universities Power Engineering Conference/Robotics and Mechatronics/Pattern Recognition Association of South Africa (SAUPEC/RobMech/PRASA)*, IEEE, 2021, pp. 1–6.

[24] N. Moslemi, R. Azmi, and M. Soryani, "Driver distraction recognition using 3d convolutional neural networks," in *2019 4th International Conference on Pattern Recognition and Image Analysis (IPRIA)*, IEEE, 2019, pp. 145–151.

[25] M. Çetinkaya and T. Acarman, "Driver activity recognition using deep learning and human pose estimation," in *2021 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, IEEE, 2021, pp. 1–5.

[26] M. Wu, X. Zhang, L. Shen, and H. Yu, "Pose-aware multi-feature fusion network for driver distraction recognition," in *2020 25th International Conference on Pattern Recognition (ICPR)*, IEEE, 2021, pp. 1228–1235.

[27] A. Ezzouhri, Z. Charouh, M. Ghogho, and Z. Guennoun, "Robust deep learning-based driver distraction detection and classification," *IEEE Access*, vol. 9, pp. 168 080–168 092, 2021.

[28] M. Leekha, M. Goswami, R. R. Shah, Y. Yin, and R. Zimmermann, "Are you paying attention? detecting distracted driving in real-time," in *2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM)*, IEEE, 2019, pp. 171–180.

[29] F. Sajid, A. R. Javed, A. Basharat, N. Kryvinska, A. Afzal, and M. Rizwan, "An efficient deep learning framework for distracted driver detection," *IEEE Access*, vol. 9, pp. 169 270–169 280, 2021.

[30] J. Wang, Z. Wu, F. Li, and J. Zhang, "A data augmentation approach to distracted driving detection," *Future internet*, vol. 13, no. 1, p. 1, 2020.

[31] Q. Xiong, J. Lin, W. Yue, S. Liu, Y. Liu, and C. Ding, "A deep learning approach to driver distraction detection of using mobile phone," in *2019 IEEE Vehicle Power and Propulsion Conference (VPPC)*, IEEE, 2019, pp. 1–5.

[32] U. Muhammad, D. R. Beddiar, and M. Oussalah, "Domain generalization via ensemble stacking for face presentation attack detection," *arXiv preprint arXiv:2301.02145*, 2023.

[33] A. Torralba and A. A. Efros, "Unbiased look at dataset bias," in *CVPR 2011*, IEEE, 2011, pp. 1521–1528.

[34] C. Duan, Z. Liu, J. Xia, M. Zhang, J. Liao, and L. Cao, "Enhancing cross-dataset performance of distracted driving detection with score-softmax classifier," *arXiv preprint arXiv:2310.05202*, 2023.

[35] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7167–7176.

[36] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 7464–7475.

[37] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, Springer, 2014, pp. 740–755.

[38] F. Zandamela, T. Ratshidaho, F. Nicolls, and G. G. Stoltz, "Cross-dataset performance evaluation of deep learning distracted driver detection algorithms," 2022.

[39] F. Zandamela, F. Nicolls, D. Kunene, and G. Stoltz, "Enhancing distracted driver detection with human body activity recognition using deep learning," *South African Journal of Industrial Engineering*, vol. 34, no. 4, pp. 1–17, 2024.

[40] M. D. Hssayeni, S. Saxena, R. Ptucha, and A. Savakis, "Distracted driver detection: Deep learning vs handcrafted features," *Electronic Imaging*, vol. 29, pp. 20–26, 2017.

[41] S. Kar, M. Bhagat, and A. Routray, "Eeg signal analysis for the assessment and quantification of driver's fatigue," *Transportation research part F: traffic psychology and behaviour*, vol. 13, no. 5, pp. 297–306, 2010.

[42] T. Öberg, "Muscle fatigue and calibration of emg measurements," *Journal of Electromyography and Kinesiology*, vol. 5, no. 4, pp. 239–243, 1995.

[43] C. Yan, H. Jiang, B. Zhang, and F. Coenen, "Recognizing driver inattention by convolutional neural networks," in *2015 8th International Congress on Image and Signal Processing (CISP)*, IEEE, 2015, pp. 680–685.

[44] C. Zhang, H. Wang, and R. Fu, "Automated detection of driver fatigue based on entropy and complexity measures," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 1, pp. 168–177, 2013.

[45] R. A. Berri, A. G. Silva, R. S. Parpinelli, E. Girardi, and R. Arthur, "A pattern recognition system for detecting use of mobile phones while driving," in *2014 International conference on computer vision theory and applications (VISAPP)*, IEEE, vol. 2, 2014, pp. 411–418.

[46] R. Laroca, E. Severo, L. A. Zanlorensi, *et al.*, "A robust real-time automatic license plate recognition based on the yolo detector," in *2018 international joint conference on neural networks (ijcnn)*, IEEE, 2018, pp. 1–10.

[47] F. Omerustaoglu, C. O. Sakar, and G. Kar, "Distracted driver detection by combining in-vehicle and image data using deep learning," *Applied Soft Computing*, vol. 96, p. 106 657, 2020.

[48] A. Esteva, B. Kuprel, R. A. Novoa, *et al.*, "Dermatologist-level classification of skin cancer with deep neural networks," *nature*, vol. 542, no. 7639, pp. 115–118, 2017.

[49] S. Frizzi, R. Kaabi, M. Bouchouicha, J.-M. Ginoux, E. Moreau, and F. Fnaiech, "Convolutional neural network for video fire and smoke detection," in *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*, IEEE, 2016, pp. 877–882.

[50] I. Goodfellow, Y. Bengio, and A. Courville, "Convolutional networks," in *Deep Learning*, MIT press, 2016, ch. 9, pp. 330–371.

[51] R. Kruse, S. Mostaghim, C. Borgelt, C. Braune, and M. Steinbrecher, "Multi-layer perceptrons," in *Computational intelligence: a methodological introduction*, Springer, 2022, pp. 53–124.

[52] J. Lederer, "Activation functions in artificial neural networks: A systematic overview," *arXiv preprint arXiv:2101.09957*, 2021.

[53] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.

[54] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *The Journal of physiology*, vol. 148, no. 3, p. 574, 1959.

[55] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[56] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[57] R. Adrian, "Convolutional neural networks," in *Deep Learning for Computer Vision with Python*, Starter Bundle, PyImageSearch, 2017, ch. 11, pp. 169–195.

[58] A. Géron, "Deep computer vision using convolutional neural networks," in *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*, N. Tache, Ed., O'Reilly Media, 2019, ch. 14, pp. 431–439.

[59] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pmlr, 2015, pp. 448–456.

[60] Q. Wang, Y. Ma, K. Zhao, and Y. Tian, "A comprehensive survey of loss functions in machine learning," *Annals of Data Science*, pp. 1–26, 2020.

[61] J. Terven, D. M. Cordova-Esparza, A. Ramirez-Pedraza, and E. A. Chavez-Urbiola, "Loss functions and metrics in deep learning. a review," *arXiv preprint arXiv:2307.02694*, 2023.

[62] I. Goodfellow, Y. Bengio, and A. Courville, "Optimization for training deep learning models," in *Deep Learning*, MIT press, 2016, ch. 8, pp. 274–294.

[63] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[64] I. Goodfellow, Y. Bengio, and A. Courville, "Machine learning basics," in *Deep Learning*, MIT press, 2016, ch. 5, pp. 98–154.

[65] J. Kukačka, V. Golkov, and D. Cremers, "Regularization for deep learning: A taxonomy," *arXiv preprint arXiv:1710.10686*, 2017.

[66] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[67] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[68] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[69] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[70] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[71] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[72] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[73] C. Li, L. Li, H. Jiang, *et al.*, "Yolov6: A single-stage object detection framework for industrial applications," *arXiv preprint arXiv:2209.02976*, 2022.

[74] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[75] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 781–10 790.

[76] M. L. Mekhalfi, C. Nicolò, Y. Bazi, M. M. Al Rahhal, N. A. Alsharif, and E. Al Maghayreh, "Contrasting yolov5, transformer, and efficientdet detectors for crop circle detection in desert," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2021.

[77] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A review of yolo algorithm developments," *Procedia Computer Science*, vol. 199, pp. 1066–1073, 2022.

[78] M. Hussain, "Yolov1 to v8: Unveiling each variant–a comprehensive review of yolo," *IEEE Access*, vol. 12, pp. 42 816–42 833, 2024.

[79] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[80] M. Hu, J. Feng, J. Hua, *et al.*, "Online convolutional re-parameterization," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 568–577.

[81] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun, "Repvgg: Making vgg-style convnets great again," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 13 733–13 742.

[82] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Artificial intelligence and statistics*, Pmlr, 2015, pp. 562–570.

[83] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A survey on performance metrics for object-detection algorithms," in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2020, pp. 237–242. DOI: 10.1109/IWSSIP48289.2020.9145130.

[84] P. Jaccard, "Étude comparative de la distribution florale dans une portion des alpes et des jura," *Bull Soc Vaudoise Sci Nat*, vol. 37, pp. 547–579, 1901.

[85] R. Adrian, "Faster r-cnns," in *Deep Learning for Computer Vision with Python*, Practitioner Bundle, PyImageSearch, 2017, ch. 15, pp. 248–250.

[86] C. Zhao, B. Zhang, J. He, and J. Lian, "Recognition of driving postures by contourlet transform and random forests," *IET Intelligent Transport Systems*, vol. 6, no. 2, pp. 161–168, 2012.

[87] C. Zhao, B. Zhang, J. Lian, J. He, T. Lin, and X. Zhang, "Classification of driving postures by support vector machines," in *2011 sixth international conference on image and graphics*, IEEE, 2011, pp. 926–930.

[88] C. H. Zhao, B. L. Zhang, X. Z. Zhang, S. Q. Zhao, and H. X. Li, "Recognition of driving postures by combined features and random subspace ensemble of multilayer perceptron classifiers," *Neural Computing and Applications*, vol. 22, pp. 175–184, 2013.

[89] C. Zhao, Y. Gao, J. He, and J. Lian, "Recognition of driving postures by multiwavelet transform and multilayer perceptron classifier," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 8, pp. 1677–1686, 2012.

[90] A. Montoya, S. Dan Holman, T. Smith, and W. Kan. "State farm distracted driver detection." Accessed on March 28, 2022. (2016), [Online]. Available: https://kaggle.com/competitions/state-farm-distracted-driver-detection.

[91] T. Billah, S. M. Rahman, M. O. Ahmad, and M. Swamy, "Recognizing distractions for assistive driving by tracking body parts," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 4, pp. 1048–1062, 2018.

[92] O. Russakovsky, J. Deng, H. Su, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, pp. 211–252, 2015.

[93] R. Sebastian, "Neural transfer learning for natural language processing," Ph.D. thesis, University of Galway, 2019.

[94] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, pp. 123–140, 1996.

[95] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, Ieee, vol. 1, 2005, pp. 886–893.

[96] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5693–5703.

[97] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7291–7299.

[98] I. Goodfellow, Y. Bengio, and A. Courville, "Sequence modeling: Recurrent and recursive nets," in *Deep Learning*, MIT press, 2016, ch. 10, pp. 330–371.

[99] Y. Xing, C. Lv, H. Wang, D. Cao, E. Velenis, and F.-Y. Wang, "Driver activity recognition for intelligent vehicles: A deep learning approach," *IEEE transactions on Vehicular Technology*, vol. 68, no. 6, pp. 5379–5390, 2019.

[100] T. Hoang Ngan Le, Y. Zheng, C. Zhu, K. Luu, and M. Savvides, "Multiple scale faster-rcnn approach to driver's cell-phone usage and hands on steering wheel detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2016, pp. 46–53.

[101] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*, PMLR, 2019, pp. 6105–6114.

[102] S. Li and W. Deng, "A deeper look at facial expression dataset bias," *IEEE Transactions on Affective Computing*, vol. 13, no. 2, pp. 881–893, 2020.

[103] A. V. Nadimpalli and A. Rattani, "On improving cross-dataset generalization of deepfake detectors," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 91–99.

[104] B. Baheti, S. Gajre, and S. Talbar, "Detection of distracted driver using convolutional neural network," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 1032–1038.

[105] A. Kashevnik, R. Shchedrin, C. Kaiser, and A. Stocker, "Driver distraction detection methods: A literature review and framework," *IEEE Access*, vol. 9, pp. 60 063–60 076, 2021.

[106] W. Li, J. Huang, G. Xie, F. Karray, and R. Li, "A survey on vision-based driver distraction analysis," *Journal of Systems Architecture*, vol. 121, p. 102 319, 2021.

[107] M. H. Saad, M. I. Khalil, and H. M. Abbas, "End-to-end driver distraction recognition using novel low lighting support dataset," in *2020 15th International Conference on Computer Engineering and Systems (ICCES)*, IEEE, 2020, pp. 1–6.

[108] J. Mafeni Mase, P. Chapman, G. P. Figueredo, and M. Torres Torres, "Benchmarking deep learning models for driver distraction detection," in *Machine Learning, Optimization, and Data Science: 6th International Conference, LOD 2020, Siena, Italy, July 19–23, 2020, Revised Selected Papers, Part II 6*, Springer, 2020, pp. 103–117.

[109] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[110] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," *Advances in neural information processing systems*, vol. 28, 2015.

[111] Y. Sasaki *et al.*, "The truth of the f-measure," *Teach tutor mater*, vol. 1, no. 5, pp. 1–5, 2007.

[112] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[113] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[114] R. Geirhos, J.-H. Jacobsen, C. Michaelis, *et al.*, "Shortcut learning in deep neural networks," *Nature Machine Intelligence*, vol. 2, no. 11, pp. 665–673, 2020.

Appendix A

Distribution of images per class in the distracted driver detection datasets

Table A.1: Distribution of images in the EZZ2021, STF, and AUC2 training sets.

Class	**EZZ2021**	**STF**	**AUC2**
C0	1681	2140	2346
C1	2255	1940	1198
C2	2363	1985	762
C3	2522	2011	641
C4	2838	1993	850
C5	2585	1980	651
C6	2430	1992	633
C7	2589	1697	591
C8	2641	1619	598
C9	2602	1816	1279
Total	24506	19173	9549

Table A.2: Distribution of images in the EZZ2021, STF, and AUC2 validation sets.

Class	**EZZ2021**	**STF**	**AUC2**
C0	100	100	100
C1	100	100	100
C2	100	100	100
C3	100	100	100
C4	100	100	100
C5	100	100	100
C6	100	100	100
C7	100	100	100
C8	100	100	100
C9	100	100	100
Total	1000	1000	1000

Table A.3: Distribution of images in the EZZ2021, STF, AUC2, CSIR test sets.

Class	**EZZ2021**	**STF**	**AUC2**	**CSIR**
C0	344	249	219	227
C1	367	227	133	84
C2	379	232	114	82
C3	374	235	99	51
C4	395	233	90	22
C5	376	232	90	13
C6	356	233	60	0
C7	381	201	63	26
C8	373	192	66	0
C9	371	213	138	5
Total	3716	2247	1072	510

Appendix B

Per-class F1-score performance results of the algorithms

Table B.1: Per-class F1-score results and overall accuracy of the ResNet50 model on the three datasets.

Class	ResNet50-EZZ2021			ResNet50-AUC2			ResNet50-STF		
	EZZ2021 test	AUC2 test	STF test	EZZ2021 test	AUC2 test	STF test	EZZ2021 test	AUC2 test	STF test
Safe driving	0.85	0.52	0.31	0.19	0.61	0.39	0.00	0.50	0.95
Text right	0.99	0.07	0.43	0.00	0.39	0.35	0.04	0.34	0.96
Talk right	0.98	0.51	0.48	0.01	0.23	0.42	0.00	0.00	0.92
Text left	1.00	0.00	0.26	0.00	0.32	0.54	0.28	0.00	0.99
Talk left	0.92	0.00	0.29	0.05	0.53	0.56	0.01	0.58	0.97
Adjust radio	0.93	0.00	0.19	0.05	0.68	0.64	0.00	0.86	0.94
Drinking	0.99	0.00	0.00	0.01	0.23	0.06	0.22	0.00	0.93
Reach behind	1.00	0.19	0.32	0.33	0.31	0.57	0.25	0.42	0.98
Make-up	0.99	0.10	0.11	0.16	0.40	0.33	0.15	0.22	0.88
Talking to passenger	0.97	0.44	0.39	0.00	0.07	0.45	0.01	0.32	0.79
Overall accuracy	96.18	27.93	31.15	16.27	40.97	44.06	16.81	36.28	99.64

Table B.2: Per-class F1-score results and overall accuracy of the EfficientNetB0 model on the three datasets.

Class	EfficientNetB0-EZZ2021			EfficientNetB0-AUC2			EfficientNetB0-STF		
	EZZ2021 test	AUC2 test	STF test	EZZ2021 test	AUC2 test	STF test	EZZ2021 test	AUC2 test	STF test
Safe driving	0.85	0.27	0.19	0.54	0.35	0.39	0.44	0.24	0.88
Text right	0.87	0.00	0.16	0.15	0.06	0.35	0.03	0.09	0.94
Talk right	0.86	0.00	0.31	0.04	0.55	0.42	0.00	0.00	0.87
Text left	0.93	0.00	0.08	0.26	0.15	0.54	0.30	0.29	0.95
Talk left	0.84	0.00	0.15	0.02	0.67	0.56	0.20	0.54	0.91
Adjust radio	0.76	0.04	0.17	0.04	0.52	0.64	0.00	0.06	0.96
Drinking	0.95	0.24	0.06	0.43	0.38	0.06	0.22	0.10	0.91
Reach behind	0.96	0.21	0.16	0.35	0.62	0.57	0.12	0.40	0.91
Make-up	0.81	0.09	0.03	0.24	0.17	0.33	0.11	0.09	0.83
Talking to passenger	0.95	0.10	0.33	0.38	0.00	0.45	0.32	0.08	0.87
Overall accuracy	87.98	13.87	17.98	26.62	34.64	43.12	18.12	15.27	90.39

Table B.3: Per-class F1-score results and overall accuracy of the convLSTM model on the three datasets.

Class	convLSTM-EZZ2021			convLSTM-AUC2			convLSTM-STF		
	EZZ2021 test	AUC2 test	STF test	EZZ2021 test	AUC2 test	STF test	EZZ2021 test	AUC2 test	STF test
Safe driving	0.97	0.00	0.00	0.45	0.04	0.11	0.00	0.23	0.97
Text right	0.99	0.00	0.09	0.18	0.10	0.04	0.16	0.21	1.00
Talk right	0.97	0.05	0.13	0.00	0.08	0.00	0.00	0.25	1.00
Text left	0.98	0.00	0.17	0.00	0.09	0.00	0.00	0.00	0.99
Talk left	0.99	0.00	0.03	0.00	0.44	0.20	0.00	0.00	1.00
Adjust radio	1.00	0.00	0.03	0.42	0.11	0.17	0.02	0.11	1.00
Drinking	0.99	0.00	0.09	0.18	0.14	0.10	0.05	0.00	0.99
Reach behind	1.00	0.02	0.08	0.19	0.39	0.32	0.08	0.28	1.00
Make-up	0.99	0.00	0.09	0.11	0.14	0.17	0.00	0.00	0.99
Talking to passenger	1.00	0.03	0.05	0.21	0.10	0.35	0.00	0.20	0.98
Overall accuracy	98.91	1.90	8.76	20.43	17.61	20.22	7.85	18.57	99.10

Table B.4: Per-class F1-score results and overall accuracy of the CNN-LSTM model on the three datasets.

Class	CNN-LSTM-EZZ2021			CNN-LSTM-AUC2			CNN-LSTM-STF		
	EZZ2021 test1	AUC2 test	STF test	EZZ2021 test	AUC2 test	STF test	EZZ2021 test	AUC2 test	STF test
Safe driving	0.62	0.04	0.00	0.13	0.39	0.22	0.15	0.09	0.87
Text right	0.89	0.00	0.00	0.02	0.00	0.00	0.03	0.10	0.99
Talk right	0.83	0.00	0.00	0.00	0.00	0.00	0.00	0.93	0.95
Text left	0.88	0.00	0.00	0.00	0.00	0.00	0.11	0.00	0.85
Talk left	0.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.96
Adjust radio	0.97	0.00	0.00	0.00	0.00	0.00	0.24	0.82	0.99
Drinking	0.94	0.00	0.00	0.00	0.00	0.32	0.00	0.07	0.98
Reach behind	0.94	0.13	0.15	0.00	0.00	0.28	0.11	0.00	0.94
Make-up	0.90	0.13	0.04	0.03	0.00	0.00	0.03	0.25	0.93
Talking to passenger	0.95	0.00	0.00	0.16	0.24	0.18	0.04	0.25	0.93
Overall accuracy	88.23	7.14	8.76	8.34	23.81	13.26	10.42	30.00	94.00

Table B.5: Per-class F1-score results and overall accuracy of the Leekha_GrabCut model on the three datasets.

Class	Leekha_GrabCut-EZZ2021			Leekha_GrabCut-AUC2			Leekha_GrabCut-STF		
	EZZ2021 test	AUC2 test	STF test	EZZ2021 test	AUC2 test	STF test	EZZ2021 test	AUC2 test	STF test
Safe driving	0.94	0.46	0.27	0.39	0.58	0.43	0.29	0.27	0.88
Text right	1.00	0.38	0.34	0.26	0.08	0.37	0.55	0.11	0.94
Talk right	0.95	0.38	0.35	0.35	0.12	0.39	0.30	0.07	0.87
Text left	0.98	0.50	0.31	0.54	0.70	0.62	0.52	0.48	0.95
Talk left	0.96	0.30	0.40	0.42	0.64	0.57	0.48	0.76	0.91
Adjust radio	0.99	0.28	0.24	0.56	0.50	0.39	0.38	0.35	0.96
Drinking	1.00	0.42	0.21	0.21	0.58	0.46	0.46	0.30	0.91
Reach behind	0.99	0.20	0.35	0.49	0.40	0.53	0.30	0.34	0.91
Make-up	0.98	0.09	0.15	0.19	0.25	0.30	0.18	0.20	0.83
Talking to passenger	0.98	0.25	0.32	0.61	0.28	0.42	0.61	0.38	0.87
Overall accuracy	97.71	32.31	30.53	40.06	44.23	43.88	42.03	33.43	88.30

Table B.6: Per-class F1-score results and overall accuracy of the CNN-Pose model on the three datasets.

Class	**CNN-Pose-EZZ2021**			**CNN-Pose-AUC2**			**CNN-Pose-STF**		
	EZZ2021 test	**AUC2 test**	**STF test**	**EZZ2021 test**	**AUC2 test**	**STF test**	**EZZ2021 test**	**AUC2 test**	**STF test**
Safe driving	0.99	0.69	0.53	0.78	0.60	0.56	0.98	0.60	0.96
Text right	1.00	0.48	0.62	0.68	0.34	0.56	1.00	0.59	0.98
Talk right	0.99	0.15	0.58	0.55	0.69	0.58	0.98	0.05	0.98
Text left	0.99	0.62	0.48	0.64	0.61	0.45	0.97	0.82	1.00
Talk left	1.00	0.99	0.89	0.68	0.69	0.79	0.99	0.89	0.99
Adjust radio	1.00	0.43	0.39	0.56	0.66	0.63	0.99	0.52	0.99
Drinking	1.00	0.63	0.19	0.52	0.50	0.27	0.97	0.18	0.98
Reach behind	0.99	0.39	0.50	0.61	0.56	0.77	0.97	0.63	1.00
Make-up	0.99	0.30	0.38	0.36	0.23	0.36	0.96	0.22	0.95
Talking to passenger	0.99	0.53	0.56	0.77	0.48	0.68	0.96	0.64	0.94
Overall accuracy	99.35	52.45	53.52	58.59	51.49	56.48	97.53	50.82	97.76

Appendix C

Class Activation Maps

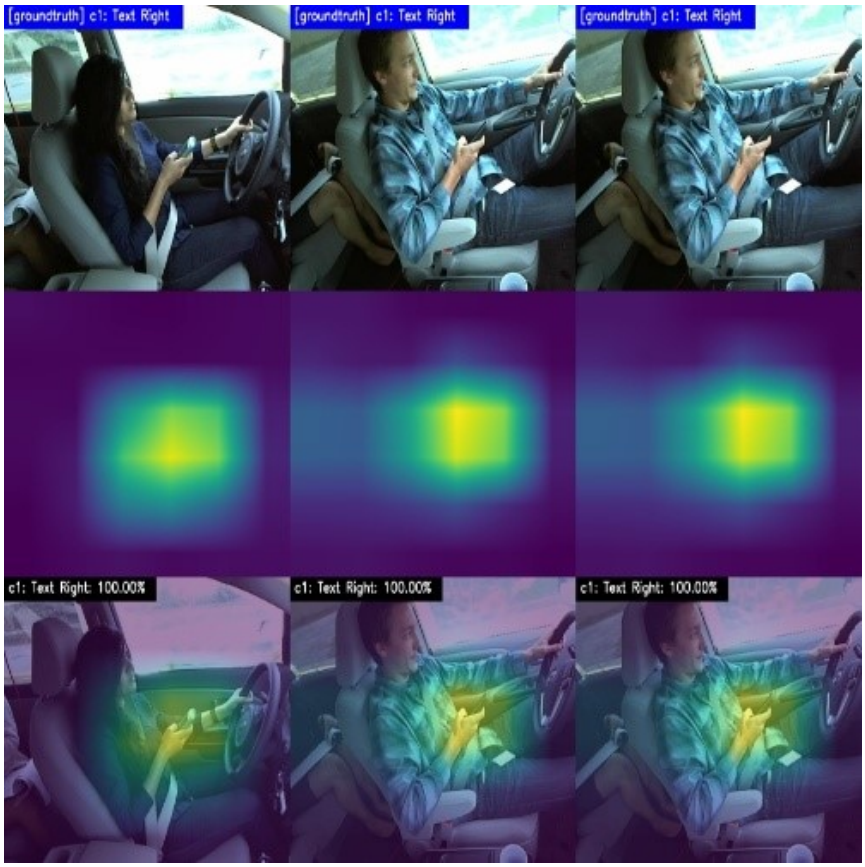


Figure C.1: Grad-CAM example for the "Text right" class.

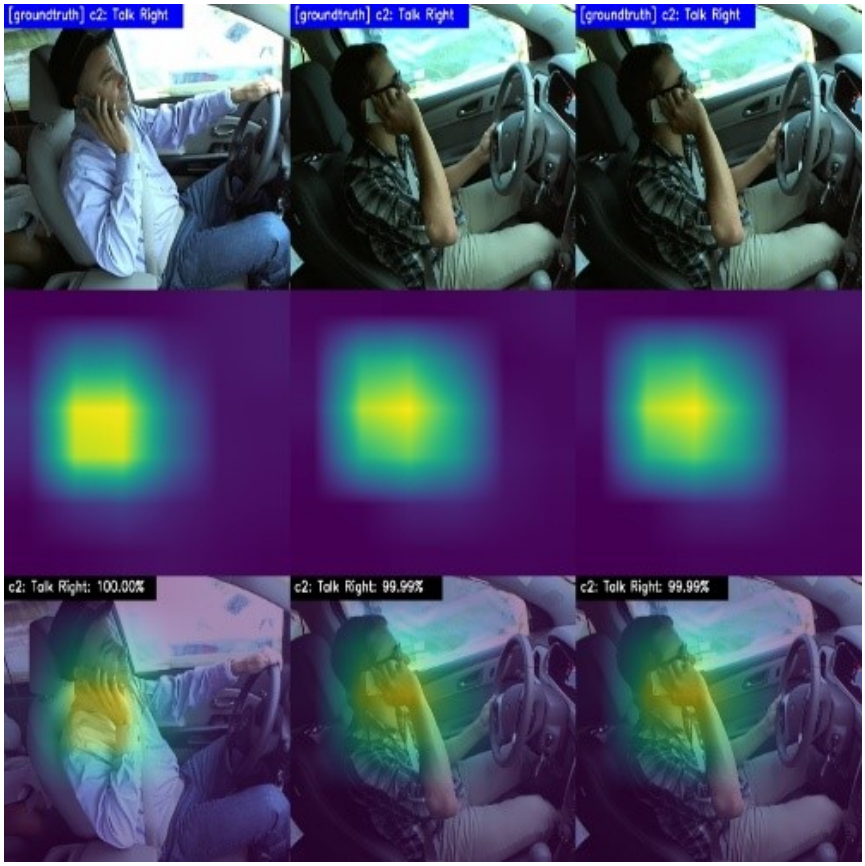


Figure C.2: Grad-CAM example for the "Talk right" class.

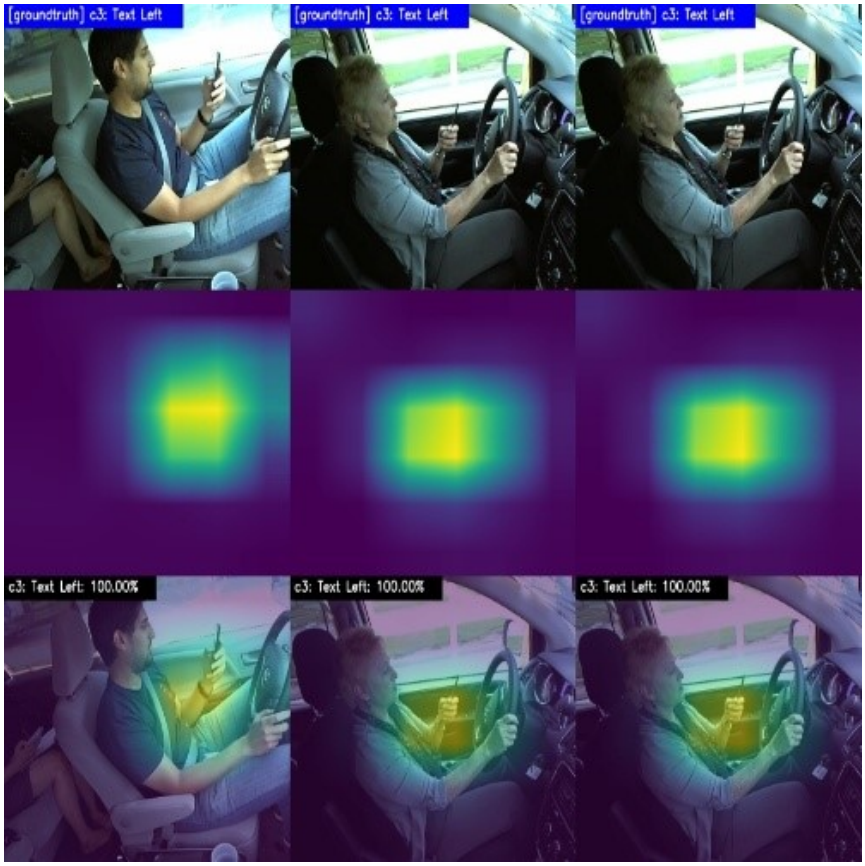


Figure C.3: Grad-CAM example for the "Text left" class.

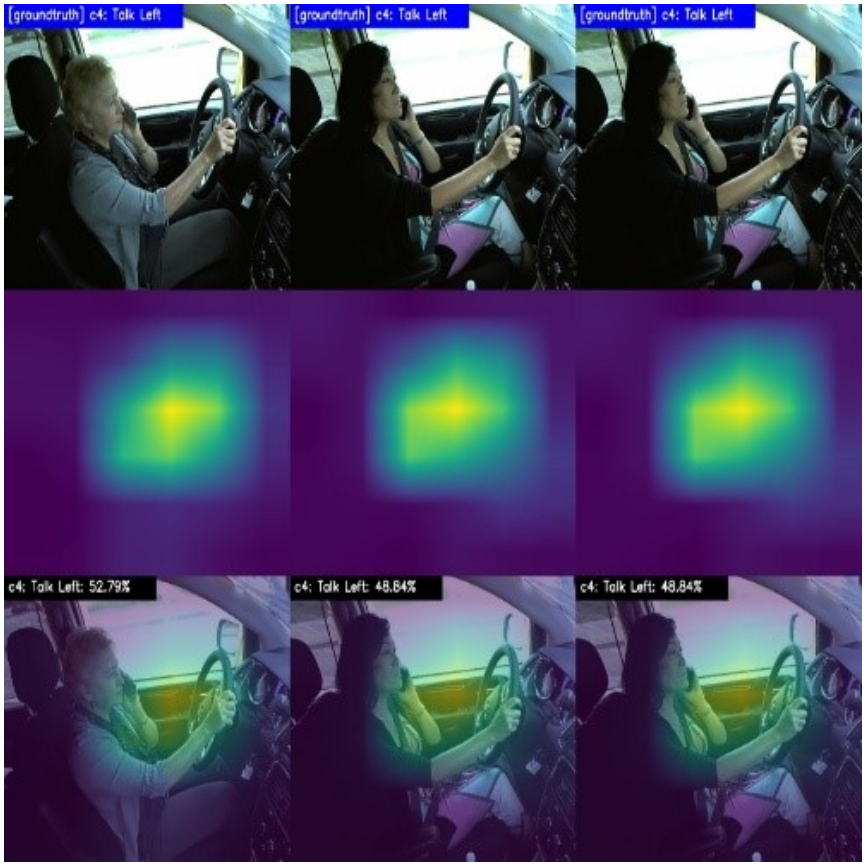


Figure C.4: Grad-CAM example for the "Talk left" class.

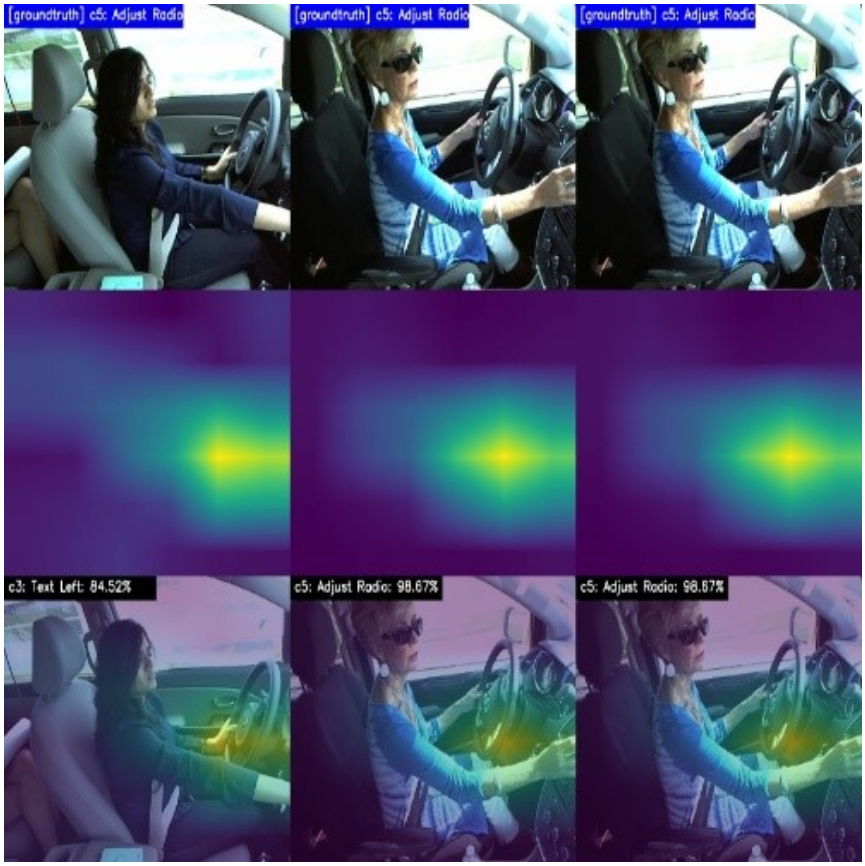


Figure C.5: Grad-CAM example for the "Adjust radio" class.

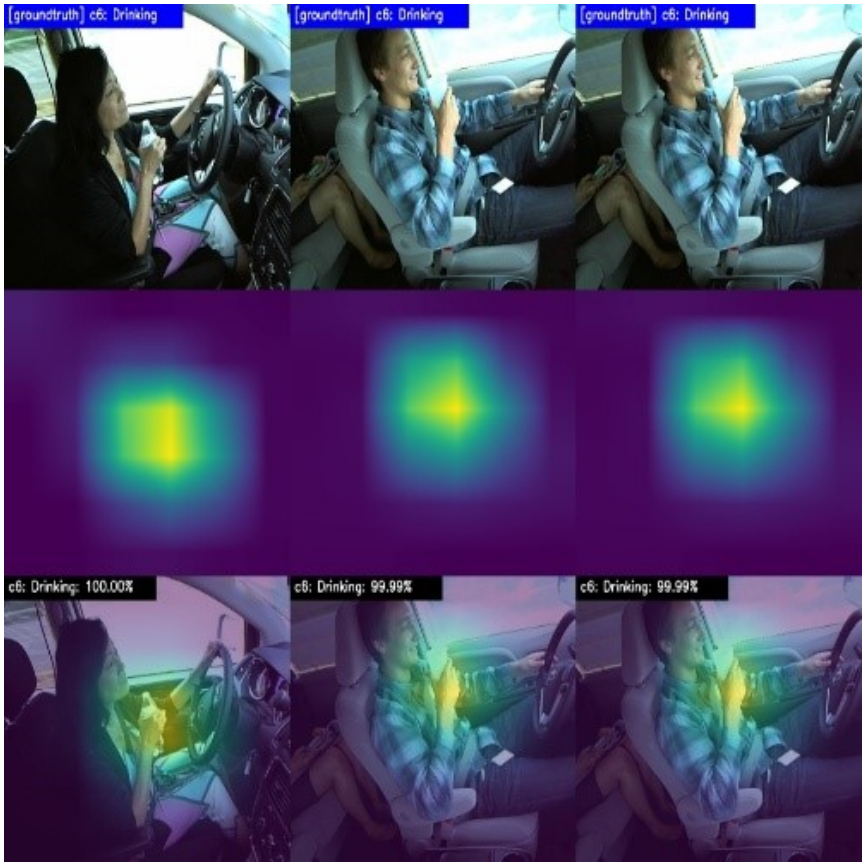


Figure C.6: Grad-CAM example for the "Drinking" class.

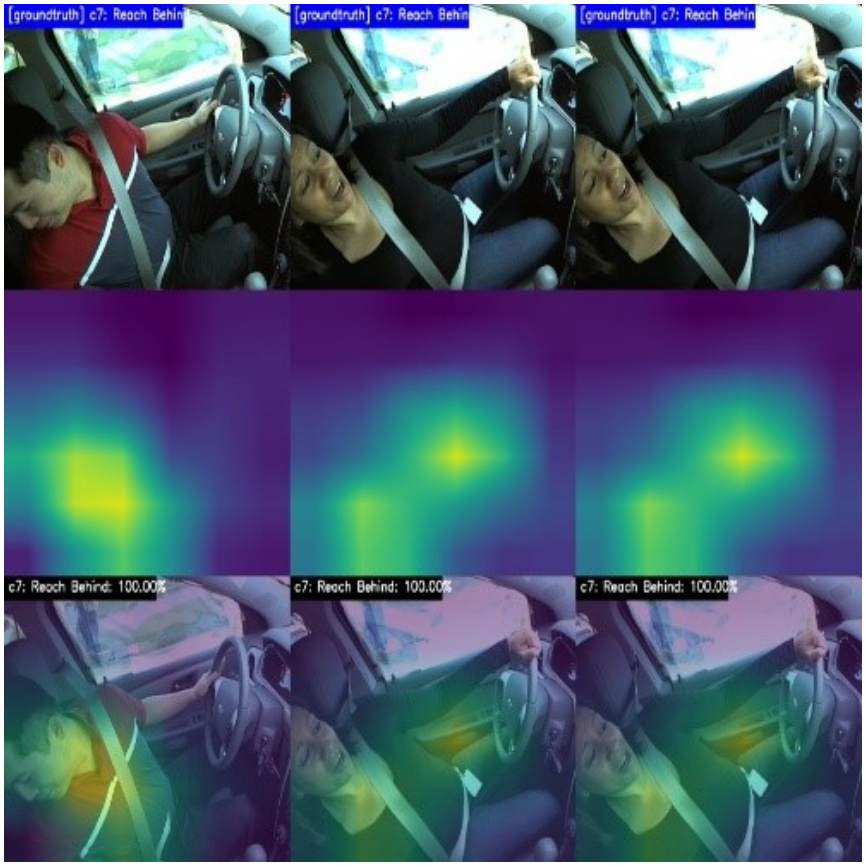


Figure C.7: Grad-CAM example for the "Reach behind" class.

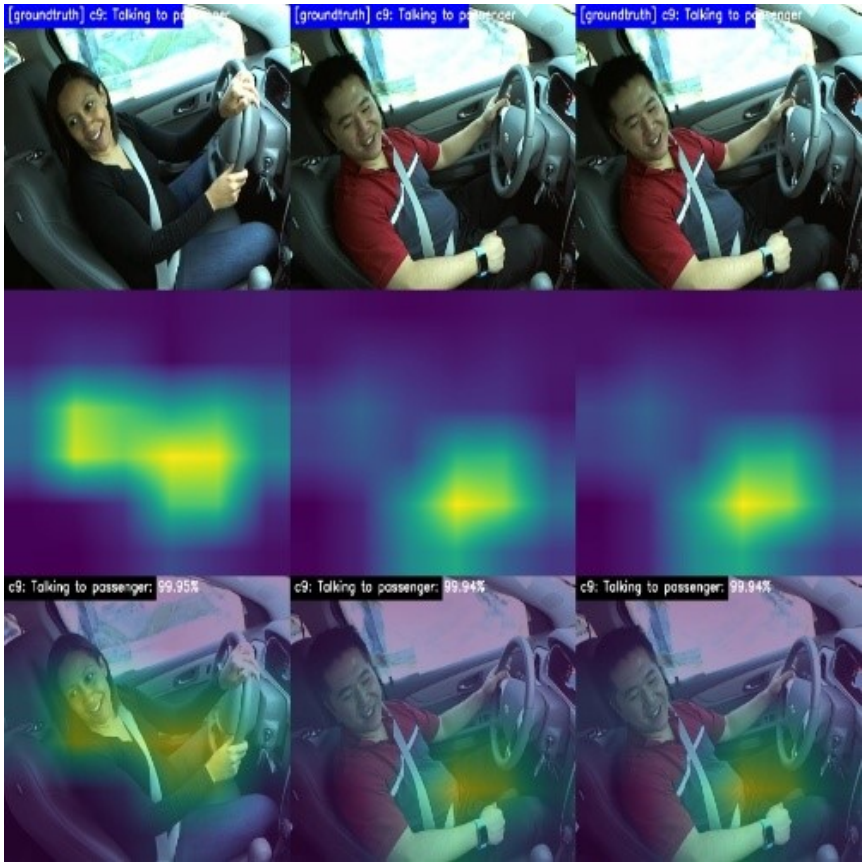


Figure C.8: Grad-CAM example for the "Talking to passenger" class.

Appendix D

Confusion matrices of the algorithms

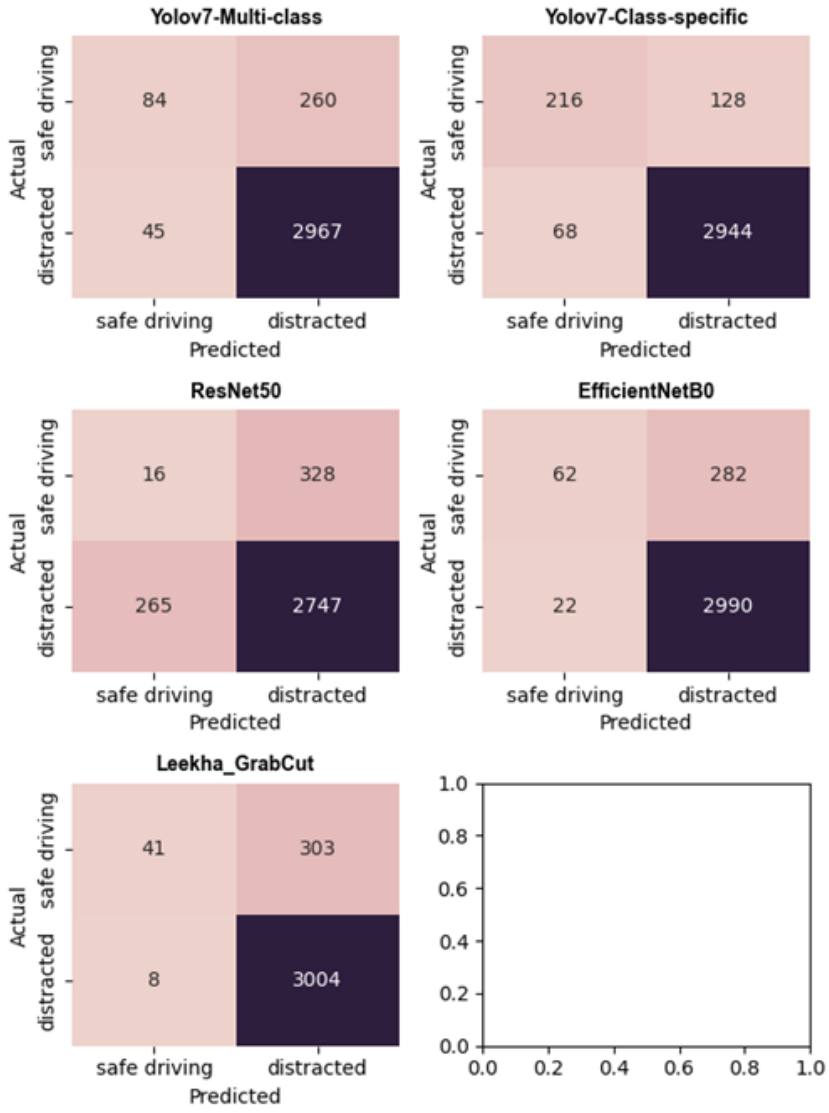


Figure D.1: Confusion matrices of the algorithms on the EZZ2021 test set.

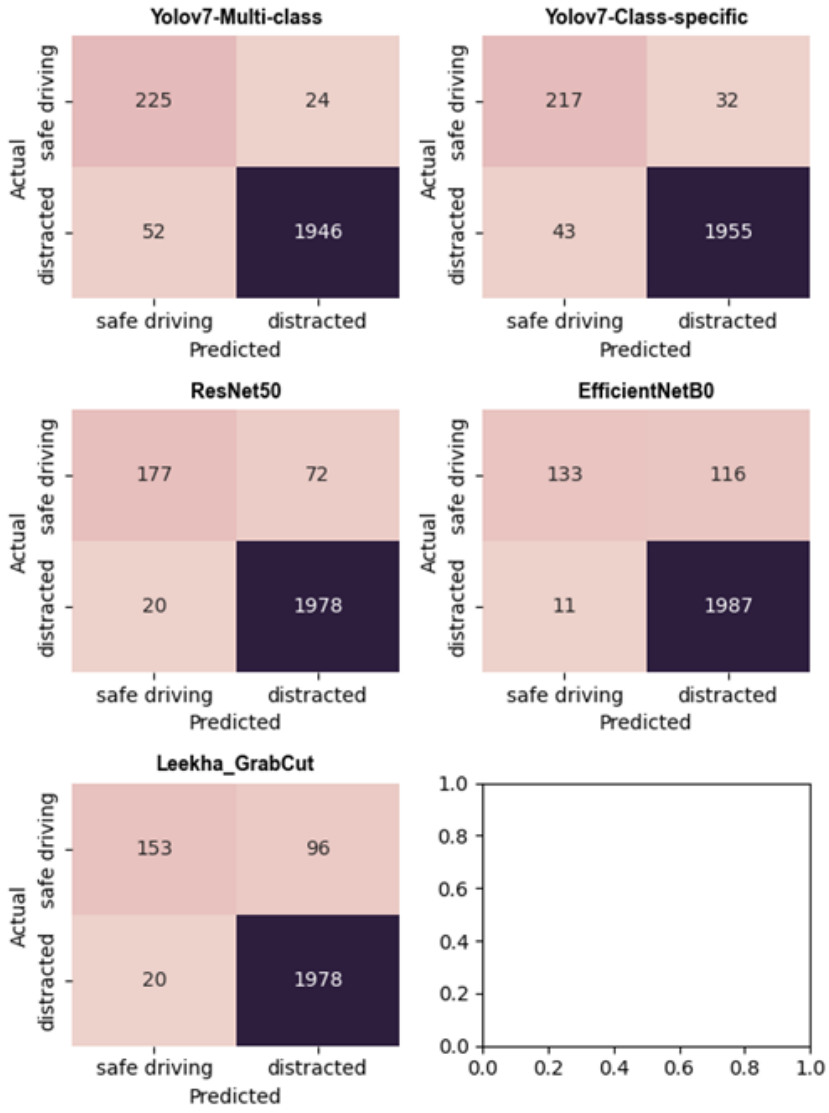


Figure D.2: Confusion matrices of the algorithms on the STF test set.

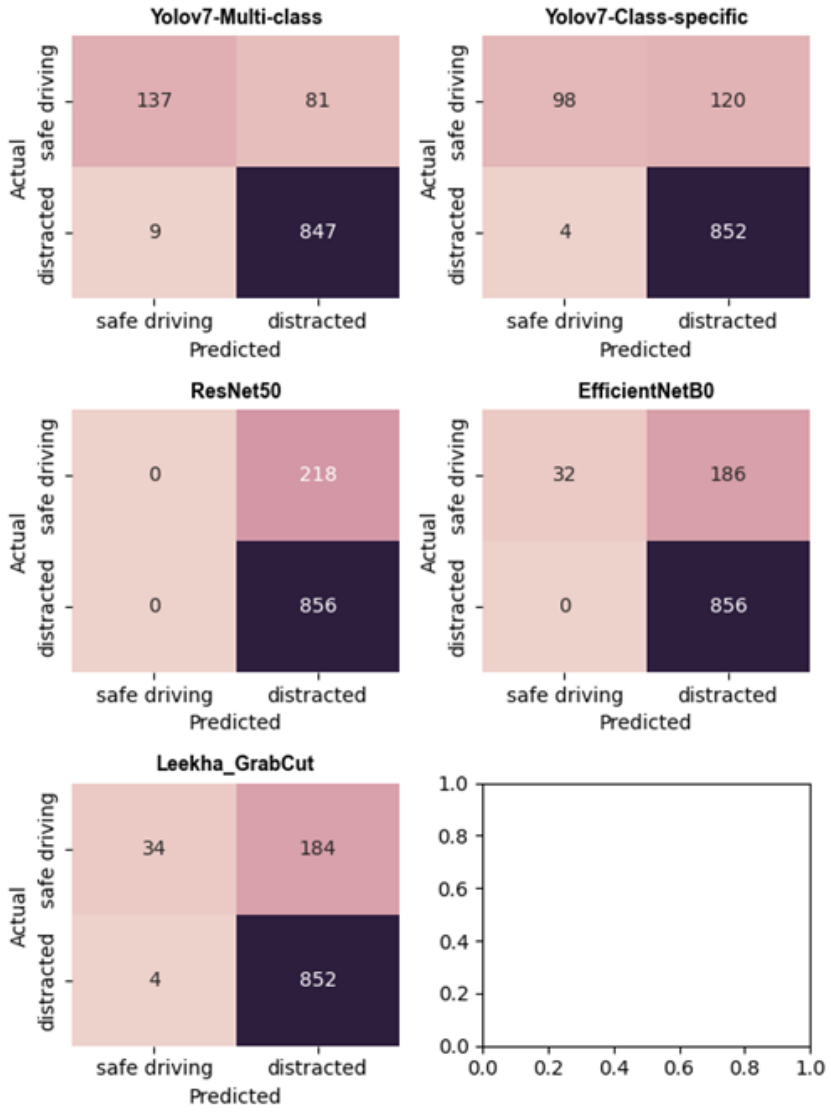


Figure D.3: Confusion matrices of the algorithms on the AUC2 test set.

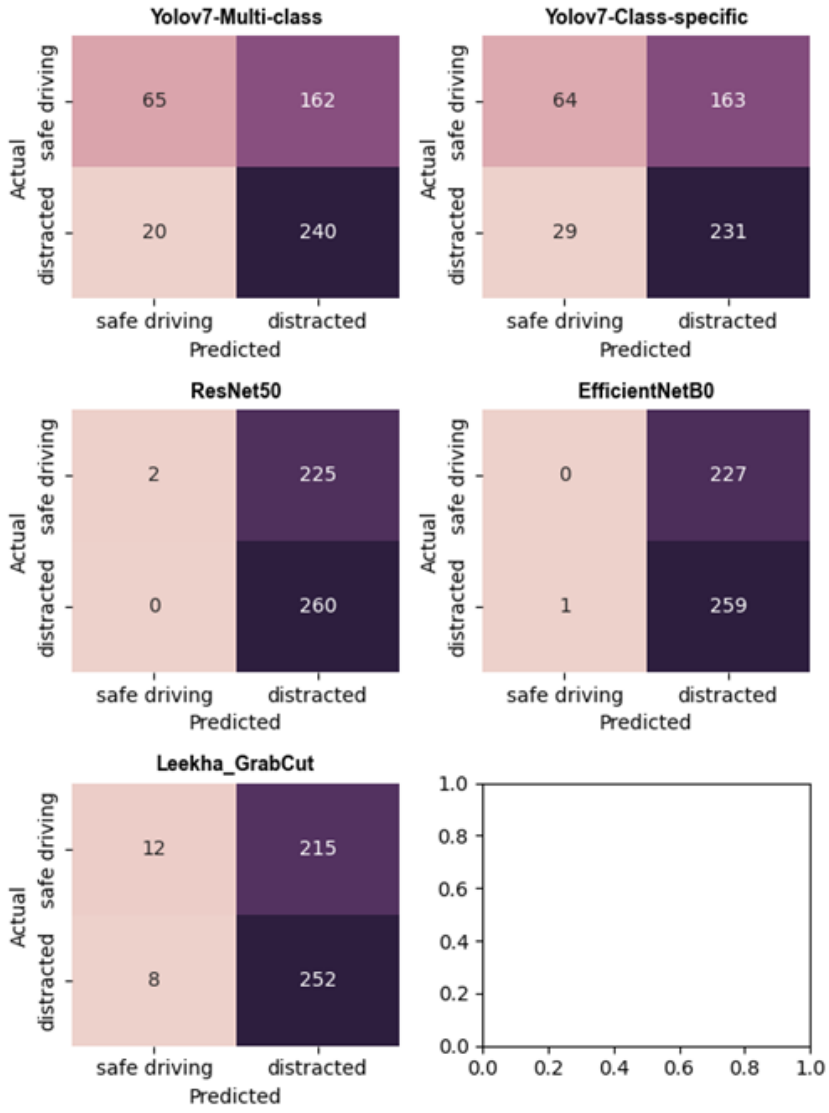


Figure D.4: Confusion matrices of the algorithms on the CSIR test set.