Deep Learning-based Driver Behavior Modeling and Analysis

by

Chaojie Ou

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

Waterloo, Ontario, Canada, 2019

Examining Committee Membership

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

External Examiner:	Prof. Simon Yang Professor, Dept. of Engineering, University of Guelph
Supervisor(s):	Prof. Fakhri Karray Dept. of Electrical and Computer Engineering University of Waterloo
Internal Member:	Prof. Mark Crowley Dept. of Electrical and Computer Engineering University of Waterloo
	Prof. Zhou Wang Dept. of Electrical and Computer Engineering University of Waterloo
Internal-External Member:	Prof. William Melek Dept. of Mechanical and Mechatronics Engineering University of Waterloo

Abstract

Driving safety continues receiving widespread attention from car designers, safety regulators, and automotive research community as driving accidents due to driver distraction or fatigue have increased drastically over the years. In the past decades, there has been a remarkable push towards designing and developing new driver assistance systems with much better recognition and prediction capabilities. Equipped with various sensory systems, these Advanced Driver Assistance Systems (ADAS) are able to accurately perceive information on road conditions, predict traffic situations, estimate driving risks, and provide drivers with imminent warnings and visual assistance. In this thesis, we focus on two main aspects of driver behavior modeling in the design of new generation of ADAS.

We first aim at improving the generalization ability of driver distraction recognition systems to diverse driving scenarios using the latest tools of machine learning and connectionist modeling, namely deep learning. To this end, we collect a large dataset of images on various driving situations of drivers from the Internet. Then we introduce Generative Adversarial Networks (GANs) as a data augmentation tool to enhance detection accuracy. A novel driver monitoring system is also introduced. This monitoring system combines multi-information resources, including a driver distraction recognition system, to assess the danger levels of driving situations. Moreover, this thesis proposes a multimodal system for distraction recognition under various lighting conditions and presents a new Convolutional Neural Network (CNN) architecture, which can operate real-time on a resources-limited computational platform. The new CNN is built upon a novel network bottleneck of Depthwise Separable Convolution layers.

The second part of this thesis focuses on driver maneuver prediction, which infers the direction a driver will turn to before a green traffic light is on and predicts accurately whether or not he/she will change the current driving lane. Here, a new method to label driving maneuver records is proposed, by which driving feature sequences for the training of prediction systems are more closely related to their labels. To this end, a new prediction system, which is based on Quasi-Recurrent Neural Networks, is introduced. In addition, and as an application of maneuver prediction, a novel driving proficiency assessment method is proposed. This method exploits the generalization abilities of different maneuver prediction systems to estimate drivers' driving abilities, and it demonstrates several advantages against existing assessment methods.

In conjunction with the theoretical contribution, a series of comprehensive experiments are conducted, and the proposed methods are assessed against state-of-the-art works. The analysis of experimental results shows the improvement of results as compared with existing techniques.

Acknowledgements

First and foremost, I would like to express my sincere thanks to Dr. Fakhri Karray for his continuous support, which starts from my first day at the University of Waterloo. Throughout the fours years of my Ph.D study, Dr. Fakhri Karray has provided massive suggestions with his incisive and profound thoughts. Dr. Karray encouraged me to explore new theories and topics in our research fields. The knowledge and experience I learned from him is the treasure that I will cherish for my whole life. I am honored to have Dr. Zhou Wang, Dr. Mark Crowley, Dr. William Melek, and Dr. Simon X. Yang on my thesis committee.

I would like to thank all the amazing group members, especially, Alaa Khatib and Arief Koesdwiady, in the Centre for Pattern Analysis & Machine Intelligence for inspiring and encouraging talks.

I would like to thank my supervisor in NUAA, Dr. Ju Jiang and his wife, Dr. Haiyan Xu. I cannot even start my study in Waterloo, not to mention to finish it.

I also wish to thank all my wonderful friends in Waterloo for cheerful time we spent together. They are Weiya Ye, Kede Ma, Wentao Liu, Zhengfang Duanmu, and many others.

I would like to thank Weiya Ye. She brings so much fun into my life. I enjoy every second I spend together with her.

Finally, I would like to thank my family, my parents and brothers, for their tremendous love and support across the Pacific Ocean.

Dedication

This is dedicated to the ones I love, especially to my mother and father.

Table of Contents

List of Figures

List of Tables

Abbreviations

ADAS	Advanced Driver Assistance Systems
AU	Action Unit
BDGAT	Bidirectional generalization ability test
BDRNN	Bidirectional Recurrent Neural Network
BGMM	Bernoulli-Gaussian Mixture Model
CNN	Convolutional Neural Network
DCP	Direction Changing Period
FID Score	Fréchet Inception Distance
FOP	Frontal Observation Period
GAN	Generative Adversarial Nets
GLTP	Green lights turn prediction
HCRF	Hidden Conditional Random Field
HMM	Hidden Markov model
IVI	In-Vehicle Infotainment

KNN K-nearest neighbors algorithm

LCP Lane change prediction

LFP Lane Following Period

LIDAR Light Detection and Ranging

LSTM Long short-term memory

MS-RCNN Multiple scale faster-RCNN

NIR Near-Infrared

NN Neural Network

OBD On-Board Diagnostics

QRNN Quasi-Recurrent Neural Network

RNN Recurrent Neural Network

SOP Surrounding Observation Period

SVM Support Vector Machine

TTI Time to an Intersection

TTS Time to steering actions

V2I Vehicle-2-infrastructure

V2V Vehicle-2-vehicle

Chapter 1

Introduction

1.1 Motivation

A recent driving study reveals that European drivers spend 10% of their driving time on secondary tasks, and the third of it is spent on mobile phones which involve extensive visual and manual distraction [6]. Another statistics shows that 90% of crashes are related to driver-related factors (i.e., driver Judgement error, impairment, fatigue, and distraction) [7]. These are just two examples among a large number of evidence that danger factors leading to accidents could come from actions inside the vehicles, which may involve other passengers or pets.

Over the past decades, there has been a remarkably increasing interest in developing driver behavior modeling methods to reduce the influence of drivers' negative behaviors on driving safety in both academia and industry. These research works can be divided into two categories. The first category is strategic research. For instance, several large-scale driving studies, which record and analyze driver behaviors without experimental control, are being carried out in North America, Europe, and Australia [8, 9, 10]. These strategic projects aim at gaining a deep understanding of driver behaviors, such as finding the relationship between accidents and road conditions and revealing the developing process of accidents by reviewing statistical information. These research works can guide legislative departments and direct the design of new Advanced Driver Assistance Systems (ADAS). The second category is tactical research, and research works in this category aim at developing new in-vehicle techniques to mitigate or prevent the effects of drivers errors. For this purpose, modeling drivers' behaviors should be treated equivalently to understanding driving surroundings in ADAS. In other words, ADAS should be able to perceive the information on driving surroundings and driver's states, reason and recognize critical traffic situations by fusing that information, and respond in real time while fulfilling drivers' demands [11, 12]. However, the fact is, while the cognitive ability of ADAS to perceive driving surroundings has increased progressively over recent years with improved techniques, such as traffic sign detection and pedestrian detection systems, the ADAS' awareness of drivers' behaviors has not progressed much [13, 14].

Recent progress in autonomous driving systems provides another promising direction to decrease human error during driving. In autonomous driving systems, perceiving or modeling the behavior of drivers is, however, still necessary. Now even the most advanced autopilot system is still in NHTSA-Level 2, which means the vehicle still requires frequent human interventions [15]. An autopilot system should be monitored continuously by a human driver. However, without an effective driver monitoring system, an autopilot system does not know whether the driver is actively participating as per the system's requirement. In Level 3, an autopilot system should be able to handle most "safety-critical functions", while the driver should intervene appropriately on request [15]. Thus an autopilot system should be aware of the driver's availability to avoid shifting the control to an inattentive driver, and the best method for this is an in-cabin driver monitoring system.

Current research on driver behavior modeling is not comprehensive, and many proposed methods are limited in scope and performance [16, 17, 18]. Meanwhile, machine learning methods, which are the cornerstones of a large part of driver modeling systems, has improved rapidly in recent years. These advanced techniques could benefit driver modeling from the aspect of performance, such as accuracy and reliability. For these reasons, we aim at developing accurate and reliable driver modeling systems using advanced and innovative approaches.

1.2 Scope and Objectives

Before defining the scope of this research work, we should first distinguish between a driver behavior prediction task from a driver behavior recognition task. In a driver behavior recognition task, a behavior may be ongoing or already finished; therefore, some parts or the entire amount of resulting information about that behavior, e.g., a driving trajectory, can be collected and used for recognition. In contrast, driver behavior prediction aims to identify a behavior that will happen in the near future, which means data on that behavior itself are not available and the prediction must be implemented with information collected before the behavior starts [19]. Accordingly, this research explores two main components of driver behavior modeling in ADAS design, including driver distraction recognition and driving maneuver prediction. Driver distraction recognition aims at detecting whether a driver is driving while using a cell-phone or In-Vehicle Infotainment (IVI) devices, and driving maneuver prediction aims at predicting a driver's next maneuver before he arrives at an intersection.

The objectives of this research are to overcome the limitations of current computer-vision-based systems, especially their limitations in hand-crafted feature extraction and generalization abilities, and to develop accurate and reliable methods using advanced and innovative approaches.

1.3 Contributions

The contributions of this thesis are threefold.

1.3.1 Enhancing Driver Distraction Recognition Using Generative Adversarial Networks

This work proposes to explore images of diverse driving scenarios from the Internet and employ generative models as a data augmentation method to improve the performance of driver distraction recognition. Moreover, a novel driver monitoring system, which integrates multiple information sources, is introduced.

1.3.2 Driver Distraction Recognition Under Various Lighting Conditions

An approach for driving distraction recognition under various lighting conditions is introduced. Unlike many existing systems that work only in the daytime, the proposed system functions with excellent performance in the daytime and nighttime by utilizing two modes. Moreover, using a novel network bottleneck, we proposed a new Convolutional Neural Network (CNN) architecture with a small footprint that provides a state-of-the-art performance in both modes. The inference speed of the proposed system on a resource-limited computing platform is evaluated, and experimental results show that the proposed network achieves the highest frame rate of inference among baseline models.

1.3.3 Deep Learning-based Driving Maneuver Prediction System

A new system that predicts a driver's next direction changing action before it happens is proposed. This work first provides a new perspective on data labeling method, with which training inputs are more semantically related to their labels, and the system after training provides better performance. Then a new maneuver prediction system based on Quasi-Recurrent Neural Networks (QRNN) is proposed. The new system is able to capture spatial and temporal features better and improve the prediction performance further. Moreover, a novel application of driver maneuver prediction, which is driving proficiency assessment by bidirectional generalization ability test, is investigated.

1.4 Thesis Outline

Chapter 2 discusses related works in the literature. It starts with a brief introduction to research improving the cognitive ability of ADAS and contrasts it with existing ADAS. The following sections then provide a brief overview of the existing work on driver distraction recognition and driving maneuver prediction.

Chapter 3 introduces methods to improve the generalization ability of distraction recognition systems by using generative models. It discusses the details of implementation and evaluates the benefits through comprehensive experiments.

Chapter 4 presents a method to develop a real-time distraction recognition system on a resource-limited computing platform. A new convolutional neural network, which is based on a novel network bottleneck, and a mechanism to handle recognition under various lighting conditions, are introduced. This system has the ability of being implemented on standalone device that drivers could use easily during driving operation (put the device on a non-disturbing location on the dashboard).

Chapter 5 provides methods to improve driver maneuver prediction. A new prediction system is presented. This chapter also investigates a new application of driver maneuver prediction.

Chapter 6 concludes with a highlight of the contributions of this work and discusses possible extensions.

Chapter 2

Literature Review

This chapter surveys related works in the literature. It describes briefly the background of improving the cognitive ability of ADAS then introduces recent and the state-of-the-art modeling approaches for driver distraction recognition research and driver maneuver prediction research.

2.1 Cognitive Abilities of Advanced Driver Assistance Systems

Several ADAS support drivers by strengthening their cognition abilities, providing warnings or suggestions in the case of driving errors, or even taking over the vehicle. For this purpose, ADAS must understand and model driving environments, vehicles, and drivers. Drivers' information needed in ADAS includes their mental and psychological states, skills/profile, etc. The required information from driving environments includes the distance from the ego-vehicle to a heading car and back car, the distance to traffic lights/signs, the existence of parallel lanes, to name a few. With this information, an assistance system can estimate ongoing driving situations and predict driving risks in the future to provide appropriate assistance. A general architecture of ADAS is illustrated in Fig. 2.1. As can be seen, the

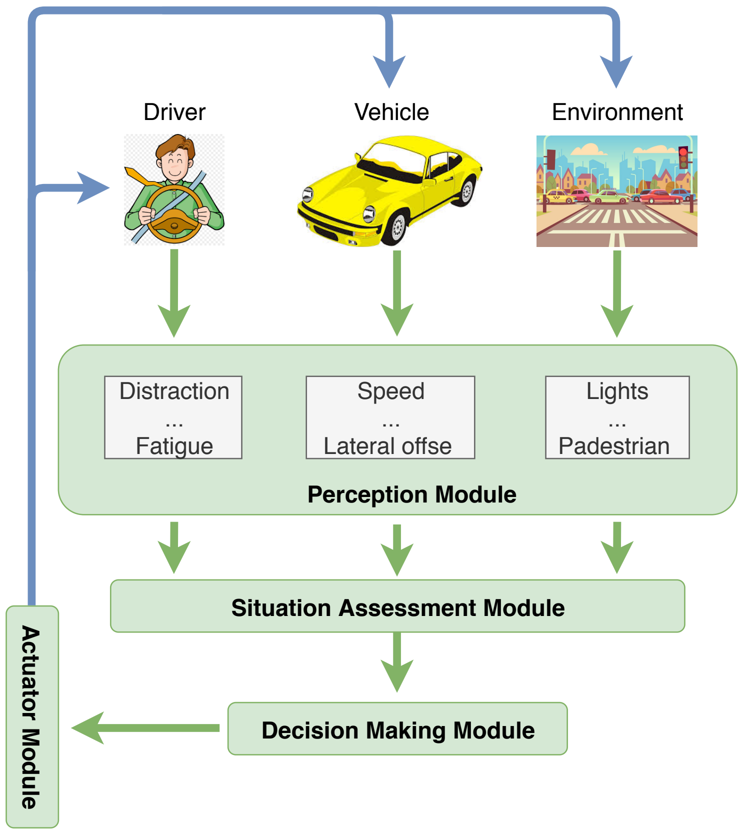


Figure 2.1: A general architecture of ADAS, which consists of four collaborative modules of corresponding functions.

architecture consists of four collaborative modules of corresponding functions. The next three paragraphs discuss these modules separately.

The Perception Module (PM) employs several different sensors and recognition systems to collect information about drivers, vehicles, and environments. A driver's information collected by the cognitive system composes a profile of him, including distraction status, fatigue status, emotional status, the proficiency of driving, the age range, etc. To implement maneuver-based driving conflict detection, the ongoing and future maneuvers of the driver are critical. Thus, the PM should also be able to recognize these two states [20]. To provide customized maneuver assistance, a driver's driving profile should also include his driving style, for example, whether he is prone to drive aggressively. The information on the vehicle includes the gear position, speed, acceleration, and some vehicle performance indexes, for example, deceleration performance is important to estimate the distance needed to brake in an emergency. This information can be extracted by analyzing data from the Controller Area Network (CAN bus) or OBD interface. The environmental information includes the lateral distance of the vehicle to lane marks, distance to the intersection, distance to the heading vehicle, distance to the rear vehicle, weather conditions, traffic light status, etc. The focus of this thesis, which includes driver distraction recognition and driving maneuver prediction, is on the PM.

The Situation Assessment Module (SAM) receives the information of the ego-vehicle and other vehicles from the PM to understand and estimate the current traffic situation, detect latent conflicts between vehicles, then send these results to the Decision Making Module (DMM) to form decisions. FOR example, the SAM MAY use information on the distance, speed, acceleration, brake information to estimate the Time-to-Collision (ToC) of the ego-vehicle to the heading vehicle when following a car. The traffic situation is an interaction between vehicles on the road, but not all vehicles are related and will affect other vehicles' actions. In front of an intersection, a driver pays more attention to vehicles that may lead to conflicts with him. For example, a driver trying to turn left should keep observing vehicles coming from the opposite direction on his left side of the road. Thus, the SAM should be able to understand the current scene and decide which vehicles are related to the ego-vehicle.

Based on results for the SAM, the DMM determines an appropriate assistance that minimizes the driving risk and improves the driving comfort. The Actuator Module (AM) implements the decision from the DMM and provides the visual display, warning, or even takes over the vehicle to execute evasive actions.

Although extensive research has been performed to improve the awareness of the Perception Module to driving surroundings and drivers, there are few commercial ADAS that possess both cognitive abilities. In the next paragraphs, we introduce several existing commercial ADAS. These systems relieve drivers from heavy cognition workloads by perceiving and reacting to the driving surroundings outside of vehicles.

Based on the distance and relative speed between vehicles, an Adaptive Cruise Control system adjusts the ego-vehicle's speed to maintain a safe distance from the vehicles ahead [21]. This system can also be extended to provide an Automatic Braking System [22]. The Automatic Braking System will alert the driver when there is a high collision risk; if the driver keeps ignoring the warning, the system will brake the vehicle automatically. Sensors involved in these systems are Radar or Light Detection and Ranging (LIDAR). The Adaptive Cruise Control system and Autonomic Braking System are both essential for autonomous driving systems in NHTSA-Level 2.

Unlike the Adaptive Cruise Control system which only controls the horizontal dynamic of vehicles, a Lane Keeping System detects the distance of the car from the lane markings and implements vertical control to prevent a lane departure [23]. With the same hardware, it is possible to develop a lane departure warning system.

An Automatic Parking System controls both the vertical and horizontal dynamics of a car to achieve parallel, perpendicular, or angular parking [24]. Basic sensors needed are Radar and cameras. With the range and parking lot mark information, the system will plan a trajectory for the vehicle, then the speed and steering angle control system will control the car to follow that trajectory.

An indispensable function of ADAS is checking the driver's state and providing adequate help. Now several car makers provide driver drowsiness detection systems which are able to detect drowsy drivers and provide warnings [17, 25, 18]. The various indicators or features used to detect driver drowsiness are: lane departure frequency, steering angle changing dynamic, driver eye closure and blinking frequency. There are also systems that detect drivers' physiological information to assess their level of drowsiness.

Another research work that incorporates driver behaviour recognition into ADAS is [16], in which the authors developed a yaw moment control system with two weighted yaw rate control strategies. The proposed control system will respond to the result of driver intention recognition and switch between two control strategies. Many researchers also believe that driver behaviour prediction can facilitate smooth and appropriate control mode transitions [26, 27]. These research works show that it is possible to integrate driver behaviour recognition and prediction systems with control systems and other on-vehicle systems to enhance driving safety and comfort [28].

2.2 Review of Driver Distraction Recognition

2.2.1 Driver Distraction Recognition Systems Design

Driver distraction is defined as “a diversion of attention away from activities critical for safe driving toward a competing activity” [29]. In [30], Klauer et al. studied this subject extensively and provided a list of distraction tasks, including fetching items in the car, using a phone, adjusting equipment on the car panel, recognizing signs outside the car, drinking, eating, and talking with other passengers in the car. Driver distraction recognition aims at detecting drivers’ engagement in secondary tasks while driving, providing assistance or generating warnings in risky driving situations. It is a fundamental problem in the design of ADAS that has not been fully resolved as of yet.

Mobile devices represent one of the most ubiquitous sources of distraction in vehicles. Many governments have taken legal steps to address the risks arising from this ubiquity, in tandem with efforts by the research community [31, 32, 33]. Overton *et al.* [34] described the danger of distracted driving, especially involving mobile devices. Having surveyed data on car accidents involving distracted driving, they found that both hands-free and hand-held devices affect driver attention negatively. In several other works, researchers used computer vision methods to monitor drivers’ heads, hands, lips, and eyes, and from there, reason about their state (whether, for example, they are talking on the phone or attending to the road) [35, 36, 37]. Signals employed by these studies pertain to the physical movements involved in distraction behaviors. Meanwhile, several other researchers take a more direct approach, which is detecting the existence of a phone near the head of a driver. Artan *et al.* [31] proposed a vision-based approach where a driver’s face is first localized using an Near-Infrared (NIR) camera system, from where features are extracted and subsequently fed to a classifier to detect a phone. Le *et al.* [38] proposed detecting a driver’s face position, hands positions, phone position, and steering wheel position using a side-view camera, and use the overlap between a driver’s hands and head to infer whether that driver is using a phone. The researchers in [39] proposed tracking relative positions of a driver's body parts and apply Support Vector Machine to classify distraction types. Other distraction activities also affect driving safety, but electronic device-related distraction has received more attention from the transportation safety departments. For instance, in Canada, Ontario's distracted driving laws apply to the use of hand-held communication/entertainment devices and certain display screens. This fact motivated this thesis to focus on electronic device usage.

Since a distraction activity is the mixture of visual, auditory, bio-mechanical, and cognitive distraction, a common topic is to distinguish only normal driving from distracted driving instead of detecting the distracting sources. A trait of these methods is that several distraction activities can be handled naturally together, and most previous works focused on this [40, 41, 42, 43].

Another problem that has been investigated is estimating the level of visual, auditory, bio-mechanical, and cognitive distraction. In [41, 43], the authors tried to find the most discriminative facial features to classify cognitive distracted driving, visually distracted driving, and normal driving. They compared the classification accuracy with different features for different distraction tasks. They found that, for visual distraction detection, gaze features and facial Action Unit (AU, represents the muscle activity that generates different facial expressions in the facial action coding system [44]) features are more discriminative, while for cognitive distraction detection, AU features are more critical. In another work conducted by the same authors, they proposed a more general method of evaluating the distraction level of drivers [42].

A large number of research works on distraction recognition are based on computer vision methods, while several researchers are trying to avoid using in-cabin cameras. They used physiological measures of drivers and indicators of driving performance to gauge driver distraction levels [45]. To detect whether a driver is texting and driving, researchers in [46] developed a system leveraging inertial sensors on the smartphone and checking the text input frequency and the frequency of typos. This system is also able to detect whether the phone user is in the car and in which position he/she is in the car. Li *et al.* proposed in [47] to detect visual-manual distracted driving by machine learning methods with features about driving performance indicators from on-board kinematic measurements. The overall accuracy achieved across all drivers is around 95% since well-designed features are utilized. Iranmanesh *et al.* used raw vehicle states from CAN as inputs to Neural Networks and Support Vector Machine (SVM) [48]. The resulting accuracy is significantly less than the accuracy from a system with well-designed features. Despite this, other experiments show that the distraction recognition system helps to reduce the miss alert rate of Forward Collision Warning system. In [49], Aksjonov *et al.* proposed a method to predict a driver's lane keeping offset and vehicle speed deviation based on road curvature and speed limit. The differences between the real and predicted lane keeping offset and vehicle speed deviation are used to infer the driver's distraction level using a fuzzy logic system.

Several research works have tried to combine various information sources to achieve better performance of distraction recognition. Addressing the detection of cognitive distraction, Liao *et al.* [50] fused the information of driving performance and eye movements. This work is a good example of exploring multi-source information (driver dynamics and visual information). Machine learning methods are widely used in distraction detection, and most of the existing works are based on supervised learning methods. Liu *et al.* proposed in [51] a semi-supervised learning method to avoid the tedious work of data labeling while enhancing the accuracy and the efficiency of distraction recognition.

The work that is most related to this thesis is in [52]. We argue that image segmentation is not necessary for the image classification in this task. Moreover, the CNNs used in [52] exhibit significant complexities thus the frame rate of the developed system will decrease significantly if deployed on a low-cost on-vehicle computing platform.

Table 2.1 summarizes several approaches in the literature to driver distraction recognition (sorted by date).

Table 2.1: Comparison of recent works on distraction recognition

Ref.	Signal type	Objective	Analysis method
[40]	Lane, head pose, car states	8 tasks	LSTM, RNN
[33]	phones, speakers	Phone use	Differential approach
[53]	Face position, mouth, hand position	Talking on the phone	HCRF
[41]	CAN-Bus, head pose, eye movement, audio	8 tasks	KNN, SVM
[31]	Image patch near head	Phone use	SVM
[54]	Arm position, eye closure, facial expression, head pose	4 tasks	Adboost, HMM, Random forest, SVM, CRF, NN
[42]	Head pose, Can-Bus, mouth expression	7 tasks	SVM, Adaboost, Random Forest
[32]	Image patch near head	Talking on the phone	Real Adaboost, SVM
[38]	Raw picture	Phone use	MS-RCNN
[55]	Raw picture	4 tasks	CNN
[56]	Raw picture and patches	10 tasks, classification	CNN

2.2.2 Design of Networks With Small Footprints

Many deep learning-based computer vision studies adopt pretrained or non-pretrained CNNs that are proposed for other applications; several widely used networks are AlexNet [57], VGG16 [58], ResNet [59], Inception [60], among others. The computation costs of these CNNs are very significant, thus hindering the usage of them in resources-limited environments. Table 2.2 presents a comparison of model sizes and numbers of parameters.

Table 2.2: Comparison of several CNN models over complexity and the number of parameters.

Model	**Size**	**Parameters**
VGG16	528 MB	138M
VGG19	549 MB	144M
ResNet50	98 MB	26M
ResNet101	171 MB	44M
MobileNet	16 MB	4M
MobileNet V2	14 MB	3M

There have been two categories of research works on developing networks with small footprints but excellent representation capabilities. The first category is network compression while the second one is designing and training a small network from scratch [61, 62, 63, 64, 65, 66, 67]. The work proposed in Chapter 4 is in the second category and is inspired by papers described in the next paragraph.

Depthwise separable convolution is introduced in [68] and is used to simply replace normal convolutional layers in [64]. In [65], the network architecture proposed in [64] is improved by employing bottleneck blocks and identity connections introduced in [59]. Instead of using depthwise separable convolution, Zhang *et al.* introduced a channel shuffle method for group convolutions and designed a CNN architecture named ShuffleNet [66]. Channel shuffle operation exchanges channels in different convolution groups so each channel has access to all information in previous layers. After exploring several guidelines and empirical studies for network design, Ma *et al.* improved ShuffleNet by adjusting the positions of shuffle layers [67].

2.3 Review of Driving Maneuver Prediction

A short survey on driver behavior prediction research is provided here, and for a more comprehensive treatment, please refer to [69] and [70].

In many research works, driver behavior prediction is also referred to as behavior anticipation, and we use in this thesis behavior prediction to maintain consistency.

Unlike driver behavior recognition, driver behavior prediction aims at anticipating a possible behavior before it happens. An early prediction gives a vehicle assistance system more time to prepare for the coming events. Various aspects of driver behavior prediction have been investigated: some researchers propose predicting drivers' next driving states which are referred to as driving maneuvers, for example turning left, or turning right [19, 71]. Meanwhile, some other researchers have focused on predicting the trajectory or other outcomes of a real steering process [72]. The ultimate aim of behavior prediction is to analyze and detect a latent driving error or risk, hence some researchers take one step forward by anticipating a potential error and risk directly [72].

Drivers' behaviors and risk predictions are particularly interesting at intersections, as traffic situations there are more complex. Drivers will encounter complex situations during interactions with other drivers and traffic signs. Accidents can be reduced if on-vehicle systems can provide enough information about other drivers' intentions and alert drivers when their decisions may lead to conflicts. For that purpose, an assistant system needs to predict the ego driver's next maneuver and other drivers' maneuvers then try to identify any conflict between them.

While evaluating a driving situation and deciding on the next action, a driver acquires information from different sources: surrounding traffic information, vehicle's states, destination, and so on. Therefore, to predict a driver's behavior, a prediction system needs all this multi-source information to be fused. Before turning right or left, drivers will check the traffic outside their cars; for acceleration, drivers move their feet first. These movements can be the hints for the next driving actions [73]. At an intersection, the layout of roads is also an important piece of information and is usually detected by an on-vehicle camera. In [74], Lefèvre *et al.* proposed exploiting digital maps on vehicles to access the topology of intersections. Other useful pieces of information for the prediction of a driver's next maneuver are the current states of his vehicle, such as the speed, steering wheel angle, etc.

Modeling methods in previous maneuver prediction research range from traditional signal processing methods to recent Machine Learning methods. To predict the imminent maneuver of the ego vehicle before an intersection, Ortiz *et al.* limited their information sources to speed, acceleration, and the distance between vehicles as these features are available both to the ego vehicle and other traffic participants [75]. With this design, it is easy to extend this system to predict other vehicles' next maneuvers and decompose a complex situation into several simple ones. Situation-specific prediction systems for different traffic situations are trained in this study. A situation-specific learned system can be used to infer the possible imminent state of a preceding vehicle, then the result of this prediction serves as an input to predict the behavior of the ego vehicle. Based on an analysis of drivers' foot behaviors, Cuong *et al.* developed a vision-based method to predict the brake and acceleration behaviors of the ego driver [76]. Through computer vision methods, they tracked the foot motion of drivers and built a Hidden Markov Model (HMM) to model the relationship between different pedal-usage states and temporal foot states. Combined with other ADAS, this system can also predict pedal errors, which means a wrong pedal is pressed or no pedal is pressed when it should be. They also showed the trade-off between the advance time and accuracy of predictions in their previous work [77].

Predicting the next maneuver of the ego vehicle is not sufficient for risk assessment because many accidents involve multiple vehicles. In other words, other vehicles' information is essential for risk assessment. However, it is difficult to collect this information and estimate other drivers' next maneuvers. There have been several attempts to estimate the states of other traffic participants, but the performances still suffer from complex changing outdoor conditions [78, 74]. In [79], the relationship between a driver's decision and vehicle states is modeled by a hybrid-state system, and the relationship between a driver's behavior and some easily observed states of other vehicles is modeled by an HMM to estimate every driver's decision. The eventual purpose of this system is to predict other drivers' decisions before an intersection is in sight. Another research work that aims at developing a framework for predicting other vehicles' maneuvers is outlined in [71]. In this work, Wiest *et al.* modeled the relationship between states of other vehicles, the intersection geometry, and the possible maneuver of the ego vehicle by a Bernoulli-Gaussian Mixture Model (BGMM). As in several other works, they tried to model that probabilistic relationship within a single model. The introduction of online learning BGMM in this work makes it possible to construct a unified model for all situations. With the development of the Vehicle-2-vehicle (V2V) communication system, the requirement to have the information on other vehicles' actions can be fulfilled by broadcasting each driver's intended maneuver among a V2V network.

Compared to red and green lights, yellow lights leave drivers with different choices, and some drivers choose to accelerate and avoid waiting. Research in [80] aims at estimating whether a driver will comply with the traffic light and stop properly. Researchers here assumed that, by a V2V or Vehicle-2-infrastructure (V2I) system, their systems can access the information about other vehicles. Part of the input to classifiers is the state series of the ego vehicle for a period before arriving at an intersection. This system also includes a sub-system to estimate the Time to an Intersection (TTI). This work explored two methods for prediction, i.e., Support Vector Machines integrated with Bayesian Filtering (SVM-BF) and HMM. For each method, this work also attempted to find the combination of features that is more discriminative. In [81], Mabuchi *et al.* developed a system to predict the decisions of other drivers when stopping or passing when the light changes to yellow. This system can also check whether a driver decides to go but stops eventually. The learning-based system takes the distance between the ego vehicle and an intersection, the speed, and the acceleration of the ego vehicle as inputs.

Lane changing happens frequently during driving, and factors affecting these actions are lane information, surrounding traffic, and drivers' intent. Lane information means which lane the ego-vehicle is in and the existence of two side lanes. The distance to other vehicles will also affect drivers' decisions, while a large distance leaves more safe space for operations. In many research works, a lane change is defined as starting when a vehicle touches a lane mark; however the vertical motion before touching a lane mark gives ample information to infer an imminent lane change, and the prediction in this period is more useful for ADAS. Simulation results in [82] show that usually it takes two seconds for a vehicle to touch a lane mark after it starts to turn. Doshi *et al.* found in [73] that head motion, lane position, and vehicle dynamic provide a good information combination to predict drivers' intended maneuvers. In addition, they found that the eye gaze estimation is more cumbersome and does not improve prediction results significantly. They also notice the obvious changing of vehicle states that happens two seconds before a lane change. These results show that lane departure is a strong sign that a lane changing may happen, while the head pose is another predictive sign. They developed an on-road test system in [83]. Ortiz *et al.* showed in [84] that it was feasible to predict the lane change of the ego vehicle with only information from a frontal and rear camera. The best accuracy they achieved was 80%; while the classification accuracy achieved in [83] was also about 80%. Kumar *et al.* tried to predict the lane change of the ego vehicle using lane information, speed, and steering angle in [85]; in their system, the probability result by an SVM is sent to a Bayesian filter for smoothing. They claimed that this system can predict the lane change an average of 1.3 seconds before it happens. In a recent study [86], many features that can be used for lane change prediction are investigated. The results show that the relative speed in relation to the front vehicle, the lateral speed to lane marks, and the distance to lane marks are the three most critical features. With these features, they predicted the possible lane change with a naive Bayes classifier, while the lane change process was modeled using a mixture of Gaussian distribution. There are also several research works that try to predict the lane changes of other vehicles to prevent possible risk. In [72], after predicting whether another vehicle will cut in in a dangerous manner, the developed system estimates the trajectory of that vehicle. The predicted trajectory, which is just an average of trajectory records of the same vehicle states, enables the ego vehicle to take corresponding evasive maneuvers.

Recent studies that are most related to our work are [87, 19, 88]. In [87, 19], Jain *et al.* developed a Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) units to predict turn and lane change maneuvers. Contextual input information is collected by multiple sensors, such as an in-vehicle camera facing the driver, an out-vehicle camera facing roads, GPS, and an On-Board Diagnostics (OBD) II to read vehicle speeds. They developed a feature fusion architecture in their network instead of just concatenating features from different sources in their previous work [89]. The memory cell in this architecture enables it to capture the long temporal dependencies between input states. Also, the depth of this network makes the representation of states more effective. The usage of RNN and multi-sources information as well as drivers' 3D head pose features promoted the performance of their system dramatically to a precision of 90.5% and a recall of 87.4% on their database. Olabiyi *et al.* extended [87, 19] by an online database collection and learning system in [88]. Moreover, they explored more features on hand positions and vehicle states for prediction. Slide-window Bidirectional Recurrent Neural Network (BDRNN) explored in this work can extract feature representations of temporal and spatial information from both input sequence ends. To make real-time predictions, their system introduces a new hyper-parameter which is the length of slide windows, and this fixed length limits the information used for predictions.

Table 2.3 summarizes several related studies on maneuver prediction from 2011. These results show that, to improve the performance of driver maneuver prediction, the new model should be able to **efficiently** fuse **multi-source** information. These information sources can be categorized into two groups: in-vehicle information sources and out-vehicle information sources. Among the in-vehicle information, the information about driver scanning behaviors, for example, the changing trajectory of facial landmarks, is critical. To obtain this information, a non-intrusive method is carried out by computer vision; however, it may be unstable with respect to frequently changing in-cabin environments. The same situation happens when collecting lane information by computer vision methods. Thus, developing a prediction system that can effectively extract and fuse visual information from both groups in a stable and robust manner is challenging but promising.

Table 2.3: Comparison of recent works on driver behavior prediction.

Ref.	**Aimed problem**	**Information**	**Analysis method**
[87, 19]	Maneuvers prediction on the road	Head pose, lane information, vehicle states	Fusion-RNN
[71]	Maneuvers at intersection	Intersection, vehicle states	BGMM
[72]	Trajectory prediction of lane change	Lateral distance to the lane mark	HMM, regression curve
[73]	Lane change, ego vehicle	Vehicles states, environment and driver state	Relevance vector machine
[90]	Lane change	Lateral velocity and position, relative velocity, time gap, time to collision	Rule-based expert system
[74]	Maneuvers at intersections	Layout of intersections, vehicle behavior	Bayesian networks
[75]	Maneuvers at intersection	Traffic lights, behavior primitives	Multilayer Perceptron
[79]	Maneuvers at intersection	Vehicle velocity, position, orientation	Hybrid-state-system, HMM
[88]	Maneuvers prediction on the road	Head pose, lane information, vehicle states	DBRNN
[85]	Lane change prediction	Lane information, speed, steering angle	SVM, Bayesian filter

2.4 Summary

This chapter provides an overview of the general architecture of ADAS within the context of preventing driver distraction and introduces several existing advanced commercial ADAS. The survey of existing work on distracted driver recognition and driving maneuver prediction shows that both of these problems are still far from solved. The next chapter will introduce our research work on enhancing driver distraction recognition using Generative Adversarial Networks.

Chapter 3

Enhancing Driver Distraction Recognition Using Generative Adversarial Networks

3.1 Introduction

Many existing works address driver distraction recognition by machine learning methods, based upon hand-crafted features representations, and this problem is formulated as a multi-class or binary-class classification task. However, feature design requires a lot of expert knowledge about signals processing and driver behavior patterns [32, 91, 92, 93]. Inspired by recent successes of CNNs in many computer vision problems [94, 95], researchers have been working on entirely data-driven end-to-end solutions for DDR [96, 55]. However, the scarcity of labeled 'real' training data stands as a significant obstacle to the efforts deployed by researchers to develop CNN-based DDR systems. Previous CNN-based DDR methods have addressed this challenge in two ways. The first approach is transfer learning. These methods adopt network architectures and parameters employed for general image classification tasks, and thus their performance is highly dependent on the closeness of the original tasks to the DDR task. The second approach is collecting images by simulation experiments on driving simulator platforms. Note that most current visual driver distraction recognition databases contain images collected by simulation experiments in simulators or real car environments [96, 55, 97, 98]. These images are extracted from video clips. Thus their backgrounds are similar, and they lie in an adjacent area within the distribution of images of distracted driving. Recognition systems trained on these datasets will overfit to those specific driving scenes and internal vehicle configurations, etc. Therefore, to ensure the effectiveness of CNN-based DDR systems on different sets of driving conditions, more diverse data would be required. However, collecting a comprehensive image dataset of distracted driving is expensive and is dangerous if it involves real driving experiments.

The work in this chapter does not images extracted from video clips for training, instead it employs an image dataset with significant diversity and explores Generative Adversarial Networks as a method for data augmentation. A dataset of images of different driving situations from the Internet is collected and used to train a Generative Adversarial Network for each driving scenario. These generative models are able to produce arbitrary numbers of images that locate closely with images of driving from the Internet in the distribution of images. Subsequently, these generated images, together with pictures from the Internet, are used to train a CNN-based distraction recognition system.

The rest of this chapter is organized as follows. The detailed framework of the proposed DDR system is described in Section 3.2. The results of experiments are presented in Section 3.3. Section 3.4 presents a driver monitoring system including a distraction recognition system as a functional module. Finally, Section 3.5 summarizes this chapter.

3.2 System Description

The proposed DDR system employs a side-view camera to acquire images of drivers in various sitting configurations. Given each image frame, a CNN will classify the state of that driver into normal driving, talking while driving, or texting while driving. The training of that CNN classifier comprises two steps. The first one is to fit a generative model for each class of driving images then sample from the image distributions of different driving scenarios. The second step is to model images of distracted driving discriminatively which is to develop a classification system. The second step makes use of the images generated in the first step. Below, these two steps are described in more details.

3.2.1 Generative Model for Additional Data Sampling

For data features (e.g., images in this work) $\{X_i^r\}$ and their labels (e.g., classes of distraction activities) $\{p_i\}$, a generative model is able to model their joint distribution. Then, by sampling from this distribution, it is possible to produce an arbitrary number of feature and label pairs $\{X_i^g, p_i\}$.

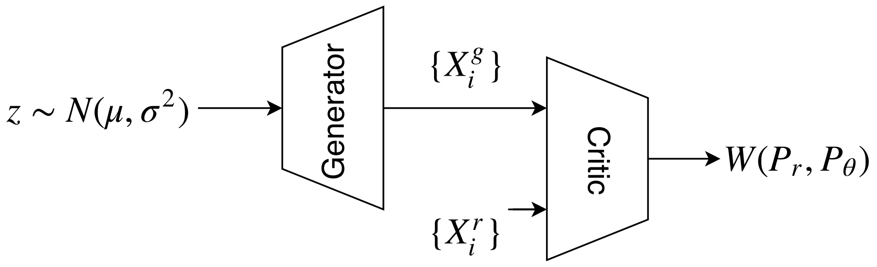


Figure 3.1: The system structure of WGANs, which consists of a Generator and a Critic.

This work uses Wasserstein Generative Adversarial Networks (WGAN) as the method to fit generative models [99]. The structure of WGAN is shown in Fig. 3.1. Let P_r be a real image distribution. The Generator is a parametric function $x = G_\theta(z)$ for a random variable z from a distribution $P(z)$, thus it defines a map from a vector distribution $P(z)$ to an image distribution $P_\theta(x)$. With a batch of image samples from P_r and a batch of image samples from $P_\theta(x)$, the Critic estimates the discrepancy between two distributions P_r and $P_\theta(x)$.

Instead of defining the discrepancy between P_r and P_θ with the *Jensen-Shannon* divergence as in the original work on GAN [100], WGAN adopts the *Earth-Mover* distance (Wasserstein-1) given as

$$W(P_r, P_\theta) = \inf_{\gamma \in \Pi(P_r, P_\theta)} E_{(x,y)\sim\gamma}\Big[\|x - y\|\Big], \tag{3.1}$$

where $\Pi(P_r, P_\theta)$ is the set of joint distributions $\gamma(x, y)$ whose marginal distributions are P_r

and P_θ. And by the *Kantorovich-Rubinstein* duality, $W(P_r, P_\theta)$ can be expressed as

$$W(P_r, P_\theta) = \sup_{\|f\|_L \leq 1} E_{x \in P_r}[f(x)] - E_{x \in P_\theta}[f(x)], \tag{3.2}$$

where the supremum is taken over all the 1-Lipschitz functions $f : \mathcal{X} \rightarrow R$. WGAN adapts neural networks with parameters w that lie in a compact set to model this critic f_w.

The training process of WGAN aims at reducing the discrepancy between the distribution P_r of training images and P_θ by learning parameter θ. The practical training procedure of WGAN contains mainly two kinds of update steps. The first one is learning f_w to maximize $E_{x \in P_r}[f_w(x)] - E_{x \in P_\theta}[f_w(x)]$, this step aims at approximating the $W(P_r, P_\theta)$. The second step is learning g_θ to minimize $-E_{x \in P_\theta}[f_w(x)]$ and reduce the discrepancy between P_r and P_θ.

After training a WGAN for each driving image class (normal driving, talking on a phone while driving, texting while driving), the Critic part is dropped. One can sample an arbitrary number of seed noise z_i from $P(z)$ then pass them through the Generator to produce new image samples X_i^g from P_θ. By this way, a dataset $S_g = \{X_i^g, C_i\}$ of generated images of different classes is composed, C_i is a one-hot indicator vector with only one entry activated to encode the ground truth distraction classes or normal driving class. Together with $S_r = \{X_i^r, C_i\}$, a new dataset of a large number of samples from the distributions of images of different driving scenes can be formed. Note this new dataset as $\{X_i, C_i\}$.

3.2.2 Discriminative Models for Distraction Classification

A discriminative model models the conditional probability distribution $P(y|x)$, so it can be used to predict y given x, where y is the label of x which may be an image or a feature combination. This work adopts CNNs as discriminative models for driving image classification. As depicted in Fig. 3.2, X_i is first fed into the feature extraction module, which transforms the raw image pixels into a feature representation optimized for distraction classification. This feature extraction module is a modified ResNet50 network from [59] for which the last fully connected layer, the average pooling layer, and the last 9 'bottlenecks' are dropped. ResNet50 is adopted in this work as it achieves a high inference frame rate on Jetson TX2 and great accuracy in our previous work [101].

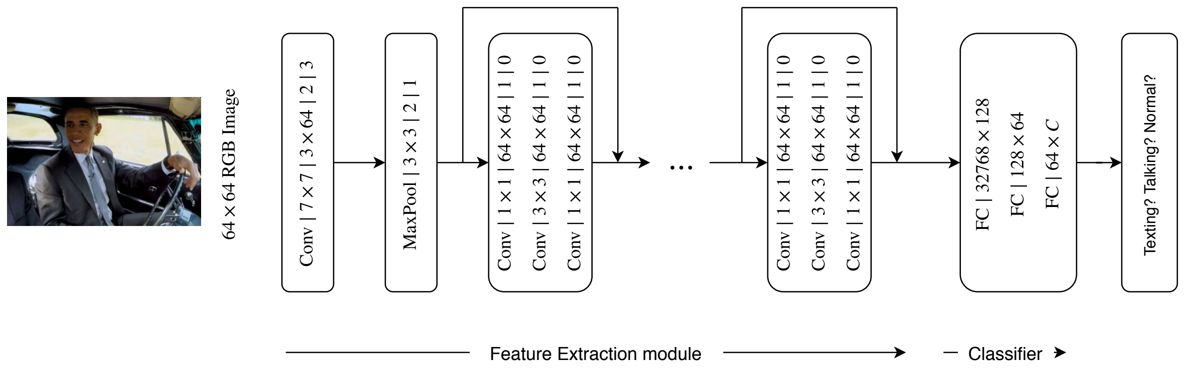

64 × 64 RGB Image
Conv | 7 × 7 | 3 × 64 | 2 | 3
MaxPool | 3 × 3 | 2 | 1
Conv | 1 × 1 | 64 × 64 | 1 | 0
Conv | 3 × 3 | 64 × 64 | 1 | 0
Conv | 1 × 1 | 64 × 64 | 1 | 0
...
Conv | 1 × 1 | 64 × 64 | 1 | 0
Conv | 3 × 3 | 64 × 64 | 1 | 0
Conv | 1 × 1 | 64 × 64 | 1 | 0
FC | 32768 × 128
FC | 128 × 64
FC | 64 × C
Texting? Talking? Normal?
Feature Extraction module
Classifier

Figure 3.2: Illustration of CNN configurations for DDR. This fig follows the convention in [1] and denotes the parameters of the convolutional layer as 'height $\times$ width | input channel $\times$ output channel | stride | padding'.

Important parameterizations of convolution, maxpooling, and connectivity for several layers are detailed in Fig. 3.2. The spatial size of images is reduced by a factor of 8 by this feature extraction module. Thus, an input image of size $64 \times 64 \times 3$ is represented by an $8 \times 8 \times 512$ dimensional feature vector. On top of the last 'bottleneck', there is a classifier module which consists of two fully connected layers with RELU activations and another softmax layer. The feature extraction module is initialized with parameters pre-trained on the ImageNet classification task, while initial parameters in the classifier module are set randomly. Learnable parameters in this feature extractor module and classifier module are collectively denoted by ψ in next sections.

The CNN classifier models the posterior probability that a given input X_i belongs to different classes, which can be expressed as

$$\hat{p}_i^k(X_i, \psi) = \frac{\exp\Big(y_i^k(X_i, \psi)\Big)}{\sum_{j=1}^{C} \exp\Big(y_i^j(X_i, \psi)\Big)}, \tag{3.3}$$

where $\hat{\mathbf{p}}_i = [\hat{p}_i^1, \cdots, \hat{p}_i^C]^T$ is a C-dimensional probability of the i-th input, which indicates the probability of each distraction class and normal driving class. An error function measures the difference between ground truth one-hot vector p_i and output $\hat{p}_i$. A widely used error function is the empirical cross entropy loss, which is

$$\mathcal{L}(\{X_i\}, \psi) = \sum_{i=1}^{N} \sum_{k=1}^{C} p_i^k \log \hat{p}_i^k (X_i, \psi). \tag{3.4}$$

The training process of CNNs is to minimize the loss function using gradient descent

$$\psi^* = \text{argmin} \mathcal{L}(\{X_i\}, \psi).$$

The partial derivative of the error with respect to each weight, $\frac{\partial \mathcal{L}}{\partial \psi}$, is calculated using the chain rule of derivatives, starting from the output of the CNN and moving backward—a process known as *backpropagation*. The gradient descent weight update performs the following iteratively

$$\psi_t = \psi_{t-1} - \eta \frac{\partial \mathcal{L}}{\partial \psi}, \tag{3.5}$$

where η is a learning rate. Given the large number of parameters they often have, CNNs are vulnerable to overfitting. A preferred practical remedy to overfitting is to add an L-2 Norm regularization term to the error function [102]. Dropout and Batch Normalization are also widely used methods to improve the generalization ability of CNNs [103, 104].

3.3 Experiment and Evaluation

This section first describes the experimental setups including details of datasets then presents three sets of experiments. In the first set, the limitation of CNN-based DDR systems trained on similar images from simulation environments is illustrated. The second set gives details of training generative models and analyzes generated image samples. The third set of experiments report the effect of generated images as a data augmentation method based on the enhancement of the detection performance.

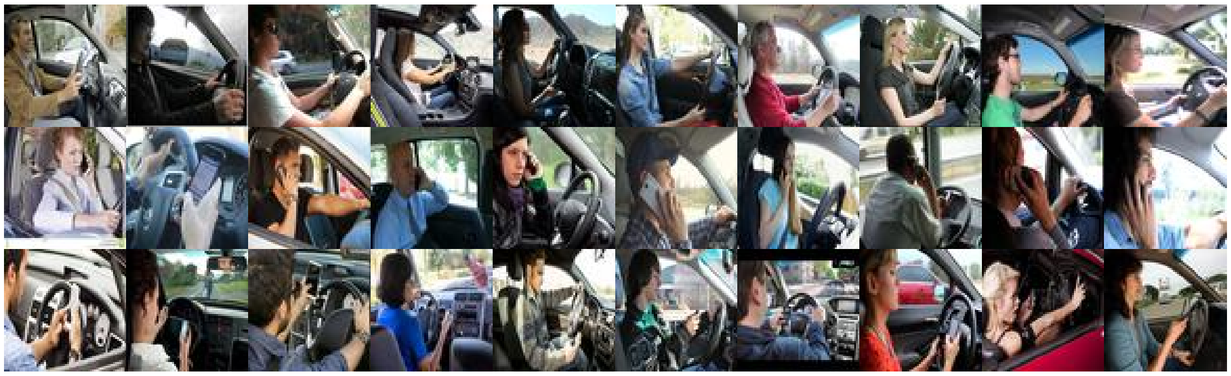


Figure 3.3: Images samples to train WGAN. The first row are images of normal driving, images of talking while driving and images of texting while driving are in the second and third row respectively. All images are resized to 64 × 64.

3.3.1 Datsets

Images Dataset of Diverse Driving Scenarios

Images collected from the simulation experiments are not diverse as to their backgrounds, drivers behavior patterns, phone models, etc. To overcome this, more than 1200 images that representing a large variety of driving scenarios are collected from the Internet. According to the manually inspection, these images represent different camera positions, image backgrounds, luminance conditions, human poses, and phone models, etc. This dataset is referred to as S_r, with representative images shown in Fig. 3.3. The numbers of images of three different classes are shown in the first column in Table 3.1.

Some images in the Internet dataset are taken from the left side of the drivers. In all our testing scenarios, however, the camera is positioned on the right side of the drivers. Thus, those images in which drivers are seen from the left are horizontally flipped. Images of frontal or back views of drivers are also excluded when training the generative model.

Table 3.1: The numbers of samples in images dataset.

Class	**Number**	
	S_r	S_s
Normal Driving	644	4600
Talking	317	9200
Texting	206	9000

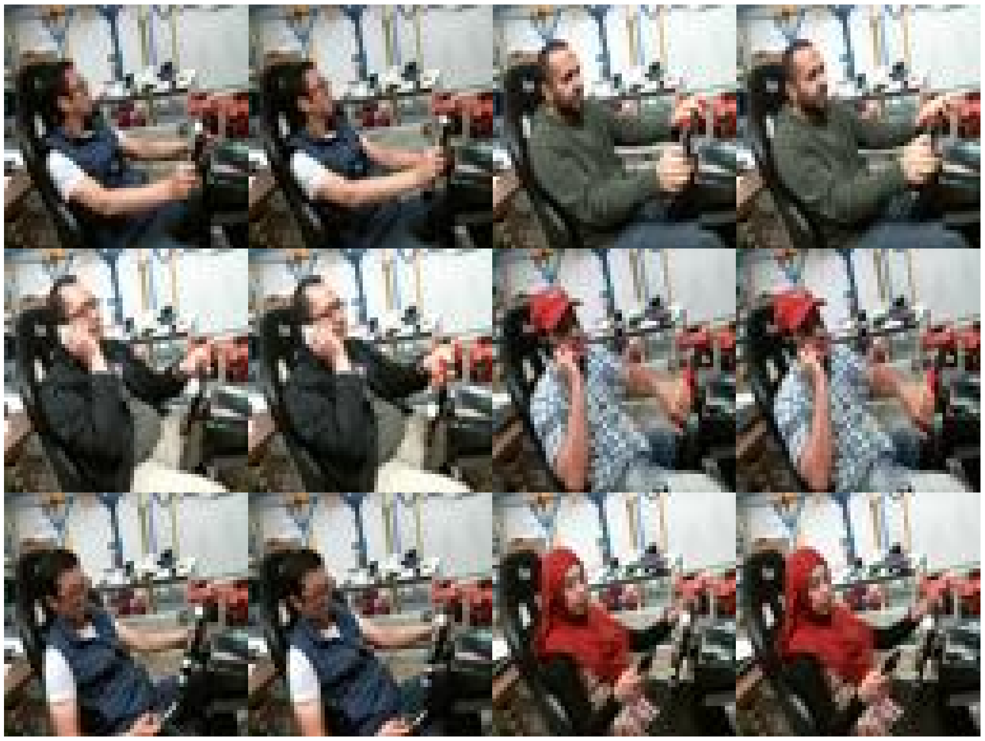


Figure 3.4: Samples of images in a simulator environment for different classes. In the first row are images of normal driving, images of talking while driving and images of texting while driving are shown in the second and third row respectively. All images are picked randomly.

Images Dataset of a Simulator Environment

Another image dataset, denoted as S_s, is collected in a simulated environment (a fixed-base driving simulator is used). Images were obtained from a camera fixed on the frontal right side of drivers under the same lighting conditions (see Fig. 3.4). There were 23 participants of different ages, genders, and ethnicities involved in this experiment. Each driver was instructed to perform the following driving activities: normal/safe driving, text messaging using the right hand while driving, phone calling using the right hand while driving. Distraction activities involving left hands are not included in this dataset since there is only a small number of images of drivers where distraction involved left hands in S_r.

3.3.2 Limitation of Training DDR Systems on Images from Video Clips

In this set of experiments, the CNN models described in 3.2.2 are trained on S_s and tested on S_r, and vice versa. These pretrained CNNs are fine-tuned with a learning rate of 1e-5, which is scaled by a factor of 0.3 after 5 epochs. The number of training epochs was 6, the batch size was 16, and no data augmentation method or weight decaying method was used during training. The same hyper-parameters were used in both training experiments for a fair comparison.

Table 3.2: Classification performance with different training and testing datasets.

Training Dataset	**Testing accuracy (%)**	
	S_r	S_s
S_r		62.61 ± 2.33
S_s	45.32 ± 1.99	

As shown in Table 3.2, the testing accuracy on S_r is only 45.32% when these CNNs are trained on S_s. On the other hand, when training on S_r, the testing accuracy on S_s is increased to 62.61%. The difference between the two testing accuracies is 17.29%. These experiments show that models trained on S_s, which is a dataset of a large amount of images extracted from a limited number of videos clips, cannot generalize well on images of more diverse driving scenes. Thus, to develop a deep learning-based driver distraction recognition system, S_r is a better training dataset. This is also supported by experiments in the next section.

3.3.3 Generate Images of Different Driving Scenarios

The classification accuracy of 62.61% achieved in the previous section by training on S_r is not enough for a distraction recognition system. One possible method to improve the accuracy is data augmentation by generative models. This section presents and analyses the second set of experiments: training a WGAN for each distracted driving class and the normal driving class, then producing an images dataset of distracted driving, denoted as S_g as in Section 3.2.1.

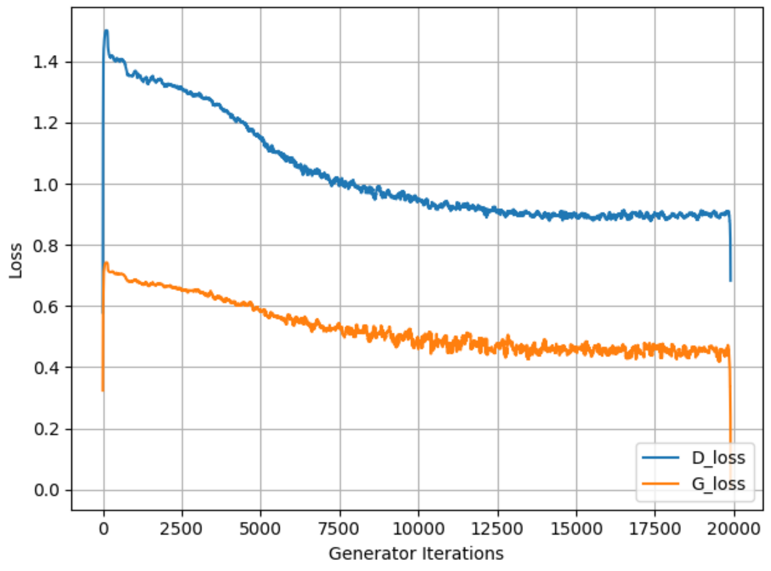
This work adopts the implementation of WGANs in [105]. For the generator in WGANs, the size of the input latent vector z is 100. The first layer in the generator is a 2D transposed convolution operator with a kernel size of 4, a stride size of 1, and a padding size of 0. The following layers are a batch normalization layer and a RELU activation layer. Treating a 2D transposed convolution layer, a batch normalization layer, and a RELU activation layer as a single module, there are another 4 modules in the generator. Kernel sizes in these four modules are all 4, stride sizes are all 2, and padding sizes are all 1. The generated images are of size 64×64.

The structure of the critic in WGANs is a mirror of the structure of the generator while all transposed convolutional layers are replaced by convolutional layers with the same kernel size, except for the last layer, which has only one output. During training, as a practical method to achieve convergence, weights w in Critic f_w are updated 8 times before updating weights θ in Generator G_θ.

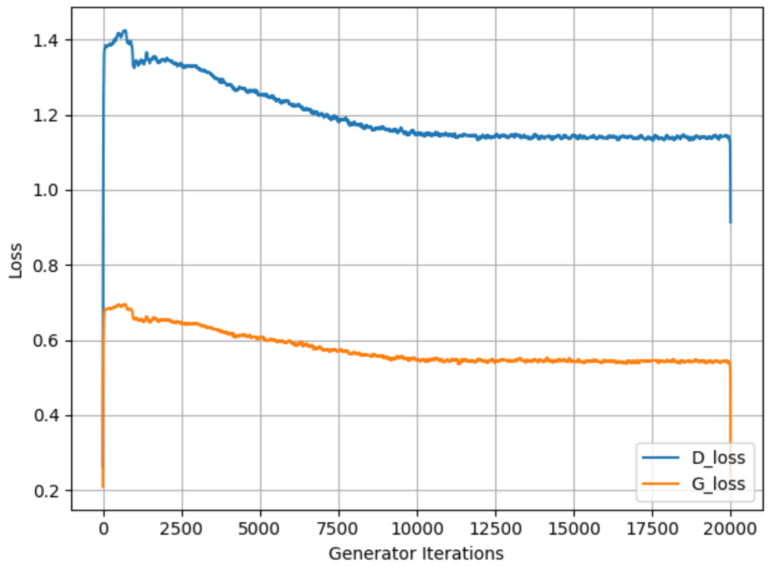
Fig. 3.5 presents three training processes of a approximate of $W(P_r, P_\theta)$ (represented by D_{loss}) and $-E_{x \in P_\theta}[f_w(x)]$ (represented by G_{loss}) on 64 samples of P_r and P_θ. These curves show results after a median filter with kernel size of 101 was applied. Fig. 3.5 (a) shows that these two losses do not change significantly after about 15000 iterations of training on images of normal driving, (G_{loss} stays at about 0.42 while D_{loss} stays at about 0.9). In Fig. 3.5 (b), these two losses stay stable at about 0.54 and 1.14 respectively after 10000 iterations of training on images of talking while driving. After about 7500 iterations of training on images of texting while driving, G_{loss} and D_{loss} stay stable at about 0.57 and 1.22 respectively as shown in Fig. 3.5 (c). According to these numbers, the training on images of normal driving takes more iterations to converge, but relatively small losses show that this training process results in a Generator that approximates closely to the real distribution of images of normal driving. Even though the numbers of images of talking while driving and texting while driving are less than the number of images of normal driving, the losses are still small enough to consider the resulting generators as adequate models of their respective distributions.

After training WGANs, one can sample from image distributions of distracted driving and normal driving by sampling latent vectors z from a Gaussian distribution then passing them through the Generators. Some samples are shown in Fig. 3.6. By checking these image samples manually, one can find that these images share many similar properties with training images. In each generated image sample of normal driving, there is a human driver sitting behind a steering wheel. In some of these images, there are two image patches whose colors are similar to the tone of human skin (labeled by red polygons). One patch is close to the upper-left corner of that image, while the other one is in the middle or close to the lower-right corner. Checking training images of normal driving, one can find that most of drivers' heads locate at the upper-left corners, while their hands are close to middle points of images. Therefore, those two patches of skin tone belong to a driver's face and two hands near the steering wheel respectively. There are also image samples that each contains three areas of human skin tone. In the 5th image in the 1st row, for example, there are two image areas close to each other that resemble two hands on a steering wheel and another image area resembles a face.

In a generated image sample of talking while driving, there are at most two areas of human skin tone, this is consistent with the fact that drivers hold their cellphones near their ears while talking, meaning the two areas of one hand and the head merge with each

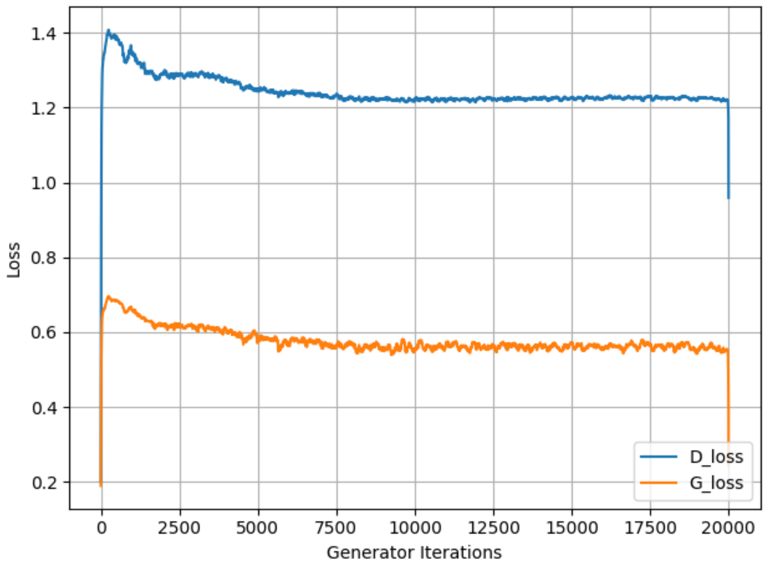


(a) Loss during training on images of normal driving



(b) Loss during training on images of talking while driving

Figure 3.5: The evolution of the WGAN during training. The decreasing process of two losses are shown here. *(cont.)*



(c) Loss during training on images of texting while driving

Figure 3.5: The evolution of the WGAN during training. The decreasing process of two losses are shown here.

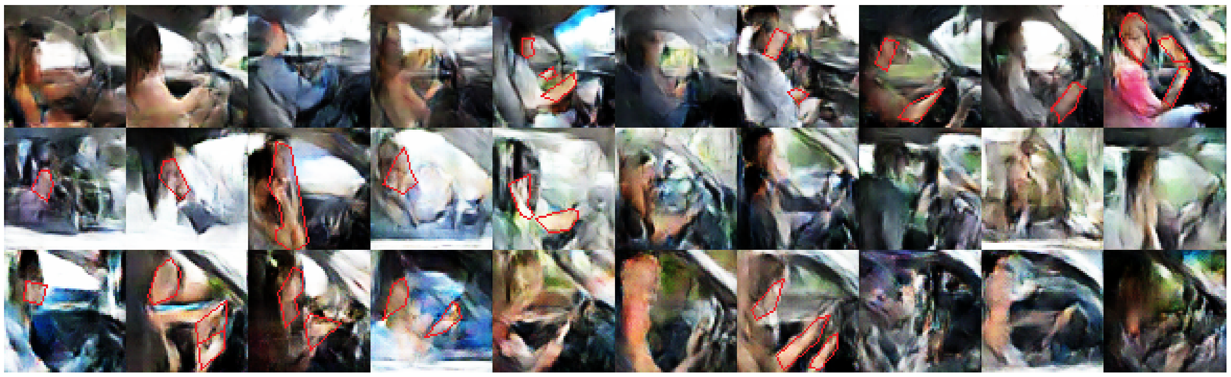


Figure 3.6: Image samples produced by sampling from distributions. In the first row are images of normal driving, images of talking while driving and images of texting while driving are shown in the second and third row respectively. All images are picked randomly.

other. For an image in which two areas of skin tone can be distinguished, the area that contains a head and a hand should be close to the left side of the image but most frequently not close to the upper corner, while the other area, which just contains a hand, is in the middle of the image.

The Generator for images of texting while driving is not trained to a very small loss, but images produced by it still show similarities to corresponding training images. In many generated image samples of texting while driving, there are three areas of human skin tone, one of them is in the upper-left and the other two are in the lower-right corner. This is because, while texting, drivers hold their cellphones near the steering wheels as shown in the third row of Fig. 3.3. The position of hands while texting and driving is very similar to that of hands during normal driving.

While the generated images are similar to the training images as to their contexts, the former do lack the details of the latter. The effect of this loss on training a distraction recognition system using these images, however, is not yet clear. For example, even though it is easy to locate cellphones by eyes in training images, generated image samples do not show clear features of cellphones. This is because WGANs, given the limited number of training images, are not able to model the fine details present in the small patches in the images corresponding to cell phones. Another issue worth noting pertains to using phones with left hands. Notice the driver in the second image of the second row of Fig. 3.3 is talking while holding her cellphone with her left hand. Left-hand drivers are rare in the training dataset, thus most produced images samples should represent right-hand drivers. But it is difficult to distinguish which hand the driver is using in a generated image since most generated images are very blurry and without clear boundary edges.

To quantitatively explore the similarity between these generated images and images used for training WGANs, this work composes a dataset S_g of generated images, trains discriminative models on S_r or S_g with a multi-class image classification task, and tests them on the other dataset respectively. The CNN described in section 3.2.2 is employed as the discriminative model. It is trained with an ADAM optimizer [106] and a learning rate of 1e-4 for 6 epochs. The training task is to classify images of three classes (Normal driving, talking while driving, and texting while driving). Testing accuracies with different training datasets and different numbers of training images are shown in Table 3.3.

Table 3.3: Testing accuracy while train on Int and test on Gen, and vice versa.

Training Dataset	**Testing Dataset**	**Accuracy (%)**
S_r	$S_g \times 6400$	92.43 ± 1.11
$S_g \times 800$	S_r	74.64 ± 0.57
$S_g \times 1600$	S_r	74.09 ± 0.67
$S_g \times 3200$	S_r	76.25 ± 1.46
$S_g \times 6400$	S_r	78.24 ± 1.59
$S_g \times 12800$	S_r	79.82 ± 1.33

This table shows that CNNs trained on S_r achieve a high average testing accuracy of 92.43% on S_g. The testing accuracy on S_r while training with 800 images from S_g is only 74.64%. With more training images, CNNs can achieve better accuracies, but only up to a point: increasing the number of training images from 6400 to 12800 only leads to a marginal increment in testing accuracies. With 12800 training images, which are more than 10 times the number of images in S_r, these CNNs can achieve an accuracy of 79.82%, which is 5.18% more than the accuracy of CNNs trained on 800 images. Accuracies of 79.82% (training on S_g and testing on S_r) and 92.43% (training on S_r and testing on S_g) suggest that the two distributions of images in S_g and S_r are very related to each other. Furthermore, the accuracy of 79.82%, which results from training on S_g with a large number of images and testing on S_r, is less than 92.43%, which is the accuracy obtained by training on S_r and testing on S_g. This shows that images in S_g cover a smaller distribution of images than S_r does. Even though the trained generative models produce images of limited diversity, the following section shows that images sampled from these generative models help to improve the performance of distraction recognition system.

3.3.4 Effect of Generated Images on Distraction Recognition Performance

The kernel part of a vision-based distraction recognition system is an image classification system which distinguishes images of distracted driving from images of normal driving.

Table 3.4: Testing performance on S_s by different training strategies.

Training Dataset	**Testing accuracy On S_s (%)**			
	Ori	L2	Aug	Aug & L2
S_r	62.61 ± 2.33	64.20 ± 2.52	68.70 ± 3.00	68.12 ± 3.91
$S_g \times 800$	61.86 ± 4.00	62.77 ± 2.26	-	-
$S_g \times 1600$	56.62 ± 1.55	55.85 ± 3.00	-	-
$S_g \times 3200$	55.49 ± 2.54	56.28 ± 4.18	-	-
$S_g \times 6400$	54.33 ± 4.17	53.88 ± 1.87	-	-
$S_g \times 12800$	51.41 ± 5.03	51.23 ± 3.94	-	-
S_r & $S_g \times 800$	74.06 ± 1.37	72.20 ± 1.85	72.63 ± 2.95	73.79 ± 1.60
S_r & $S_g \times 1600$	74.04 ± 1.85	73.03 ± 2.84	72.35 ± 3.07	72.96 ± 2.98
S_r & $S_g \times 3200$	71.74 ± 2.90	70.32 ± 2.50	72.03 ± 1.75	70.16 ± 2.92
S_r & $S_g \times 6400$	67.06 ± 4.34	61.85 ± 5.00	69.63 ± 2.10	69.28 ± 4.16
S_r & $S_g \times 12800$	63.38 ± 4.14	65.66 ± 4.11	64.40 ± 4.55	68.44 ± 4.34

The third set of experiments addresses this by employing the developed generative models to increase the accuracy of the CNN-based end-to-end image classification system.

With S_g and S_r, the CNNs described in section 3.2.2 are trained to classify images into normal driving, talking while driving, and texting while driving. Moreover, these re-trained CNNs are tested on S_s, which is an image dataset of distracted driving in a simulated environment. The training process uses an Adam optimizer with a learning rate of 1e-4 and trained CNNs for 6 epochs. The performance of these experiments is evaluated using accuracy, as shown in Table 3.4. 'L2' in this table indicates that weight decay (L2 Norm of parameters) is used during training for results of that column, while 'Aug' represents augmentation methods (i.e., random crop, random affine transform, color jitter, and random perspective transform) are used on images from S_r.

As shown in the first row of this table, augmentation methods and weigh decay can obviously improve the generalization ability of models trained on images from the Internet. With weights decay, the accuracy is increased to 64.20%±2.52% from 62.61%±2.33%; with data augmentation method, the accuracy is increased to 68.70%±3.00%; with these two methods, the accuracy is increased to 68.12%±3.91%. These results indicate that CNNs trained on S_r are able to classify images in S_s but only with limited performance, since the background environment of images in S_s is a simulated driving environment and is different from real driving environments of images in S_r.

Rows 2–6 of the first column in this table show that CNNs trained with 800 images from S_g achieve a similar average accuracy to CNNs trained on S_r. However increasing the number of training images in S_g does not introduce an increment of the testing performance; on the contrary, the testing accuracy decreases from 61.86% to 51.41% when the number of training images increases from 800 to 12800. This decremental trend suggests that training CNNs with an excessive number of images from S_g is similar to training CNNs with a small number of images but with a large number of epochs since the diversity of images in S_g is less than that of images in S_r. Furthermore, training with an excessive number of images in S_g leads to over-fitting with respect to S_s. Rows 7–11 of the first column show testing results of CNNs trained on a mixture of S_r and different numbers of images in S_g. With S_r and 800 images in S_g, CNNs achieve an average testing accuracy of 74.06% which is 11.45% more than the testing accuracy of CNNs trained on S_r only and 13.42% more than the testing accuracy of CNNs trained on 800 images in S_g. Increasing the number of training images from S_g to 3200 and more leads to a decrement of the testing performance.

Comparing results in the first row and first column in this table, one can find that, generated images help to enhance the generalization ability of trained models further, which is 5.94% more the accuracy by training on augmented S_r with weight decay. However, the application of traditional augmentation methods and weight decay on the dataset of real and generated images does not improve the accuracy further, as accuracies shown in the 7-th rows are almost the same.

The relation between the quality of generate images and the increment of generalization ability is investigated in Table 3.5. In this table, Fréchet Inception Distance (FID Score) measures the quality of generated images by the similarity between generated images and images used to train the GAN and a small value indicates better generated images [107]. The FID score is calculated per class. The second column shows results by the same GAN architecture from the first column but trained with only a quarter of its epochs, and the

Table 3.5: Comparison of testing accuracy by generated images of different qualities.

		WGAN	**DWGAN**	**CWGANGP**
FID	Normal Driving	237.79	285.22	258.10
	Talking	294.65	342.62	284.29
	Texting	315.21	327.19	288.98
ACCURACY	S_r & $S_g \times 800$	74.06 ± 1.37	63.59 ± 2.49	63.10 ± 2.14
	S_r & $S_g \times 1600$	74.04 ± 1.85	59.95 ± 3.31	60.43 ± 2.68
	S_r & $S_g \times 3200$	71.74 ± 2.90	63.49 ± 3.52	58.07 ± 2.89
	S_r & $S_g \times 6400$	67.06 ± 4.34	57.89 ± 6.33	57.07 ± 3.25
	S_r & $S_g \times 12800$	63.38 ± 4.14	58.72 ± 6.04	51.91 ± 5.50

third column shows results by the GAN model from [108] but conditioned on class labels, as shown in Fig. 3.7. One could notice that the Critic trained in these GANs are not discriminate models required in the second stage.

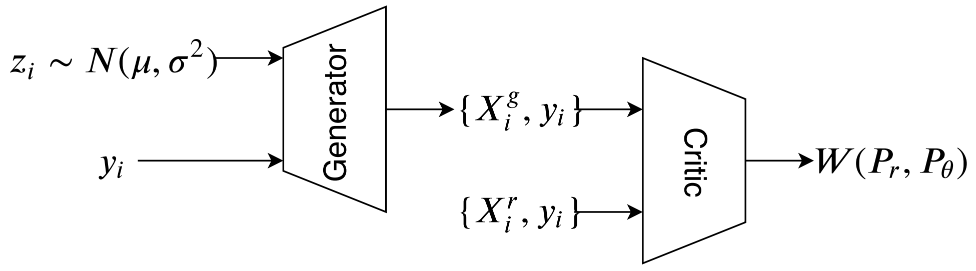


Figure 3.7: The system structure of conditional WGANs, in which the generator is conditioned on label y_i. In this structure, y_i will be concatenated with z_i to form the input to the Generator.

Two key observations from this table are as follows. First, if the WGAN is not trained with enough epochs, the similarity between training and generated images is small. Training with these images of a low quality, all training strategies achieve low accuracies. Second, the conditional GAN provides images of the best quality. However, these images of a high quality do not improve the testing accuracy for any strategies. The reason is that the FID measures perceptual quality of generated images and the same network parameters are used to generate images of different classes in the conditional WGAN.

3.4 Driver Distraction Recognition as a Module of a Driver Monitoring System

This section describes the structure of a Driver Monitoring System composed of three main modules: a driver head pose estimation module, a distraction activity recognition module, and a danger level inference module. Fig. 3.8 illustrates an overview of the proposed system.

With the driver's facial image captured by Dash Cam 1, the head pose estimation module calculates the driver head's yaw and pitch angles. If one of these two angles passes a threshold, the system decides that the driver is not focusing on the road and starts calculating the time duration t_L that he is distracted. With the driver's side view image captured by Dash Cam 2, a deep learning-based distraction recognition module estimates the probability the driver is driving safely or is distracted (i.e. talking or texting on the phone). The system calculates the time duration t_D during which the driver is distracted. With t_L and t_D as inputs, the danger level inference module infers the danger level by a fuzzy-logic-based inference mechanism. The speed of the vehicle obtained by the OBD II affects the sensitivity of the danger level inference module; while with a high speed, the danger level inference module is more sensitive to the increase in t_D and t_L.

3.4.1 Head Pose Estimation Module

The aim of this module is to determine how a driver's head is tilted with respect to the dash camera main axis. This information is an important visual cue in deciding whether the driver is paying attention to the road ahead or is oriented to a different target (e.g. radio system). This module utilizes OpenFace [109], which is an open source toolkit that provides functions to perform real-time facial behavior analysis including facial landmarks detection, gaze estimation, and head pose estimation.

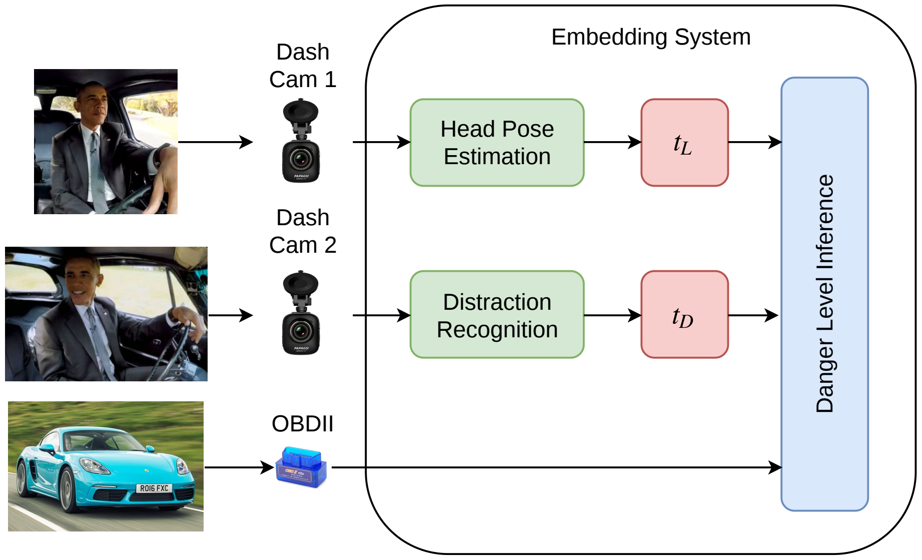


Figure 3.8: Overview of the proposed driver behavior monitoring system. Based on the driver's facial image captured via Dash Cam 1, the head pose estimation module determines the driver's head orientation. The driver's side view image is captured via Dash Cam 2 and goes through the distraction recognition module. The duration time of distraction is calculated when one of these two modules is activated. Finally, the danger level inference module estimates the danger level based of the type of distraction together with the speed of the vehicle calculated by the OBD II.

Briefly, OpenFace extracts facial landmarks from each video image containing human face using the Conditional Local Neural Fields (CLNF) proposed in [110]. Then based on the camera parameters (focal length and principal point), a 3D representation of the extracted facial landmarks is projected to 2D coordinates on the image using orthographic camera projection, allowing an accurate estimation of the head pose [110]. Our use of OpenFace is only limited to the generation of the pitch and yaw of the driver's head. This information is used later as inputs to the danger level inference module.

3.4.2 Distraction Recognition Module

The distraction recognition module is a deep learning-based system that takes as input the drivers' side view image captured by a dashcam. The aim of this system is to classify the image according to the driving behavior (e.g. normal driving, texting while driving, etc.). Hence, the driver distraction recognition problem is treated as a multi-class classification problem, which maps the input observations to distraction activities or normal driving behavior. At every time point i, the distraction recognition system receives an image I_i as input to the classifier. This latter is a deep convolutional neural network that computes the probability distribution

$$Act_i = [Act_{i1},\ Act_{i2},\ \ldots,\ Act_{iK}], \tag{3.6}$$

where Act_{iK} represents the probability that at time point i the driver is performing a distraction activity K. Moreover, $\sum_{j=1}^{K} Act_{ij} = 1$ since Act_i is a probability distribution. In other words, the classifier models the conditional probability

$$Act_{ij} = P(Act_j|I_i) = \frac{P(I_i|Act_j)P(Act_j)}{p(I_i)},\ j\ \in \{1, \ldots, K\}. \tag{3.7}$$

This module adopts the ResNet-50 pre-trained on ImageNet dataset in Keras. Imaget-Net [111] is a publicly available dataset containing more than 1.2 million images grouped into 1000 classes. In order to adapt ResNet-50 to our distraction recognition problem, the fully connected layers in the original ResNet-50 model are replaced by two fully connected layers and a softmax layer, which is commonly used in the literature [94]. These two new fully connected layers have 256 and 128 neurons respectively, rectified linear unit activation functions, and randomly initialized parameters.

3.4.3 Danger Level Inference Module

For each frame from Dash Cam 1, the head pose estimation module estimates the yaw and pitch of the driver head. If one of these two angles exceeds a threshold, the system starts calculating the number of frames that the driver is not watching the front, this number is the time period t_L. For each frame from Dash Cam 2, the distraction recognition module estimates the probability $P(Act_j|I_i)$ that the driver is performing distraction tasks. If this probability exceeds a threshold, the system starts calculating the number of frames that the driver is distracted, this number is the time period t_D.

The danger level inference module is based on a fuzzy logic approach and consists of two inputs: *HeadPose* and *Distraction* obtained from the head pose estimation and distraction recognition modules, respectively. In order to take the speed of the vehicle into consideration, the two inputs are computed as follows, where S is a rate that is propositional to the vehicle speed and is defined empirically:

$$\begin{aligned} HeadPose &= S * t_L, \\ Distraction &= S * t_D. \end{aligned} \tag{3.8}$$

We have defined the same fuzzy logic topology for both inputs. The fuzzy set is composed of three linguistic variables: *short* (between 0 and 3), *medium* (between 1 and 5) and *long* (between 3 and 10). A triangular membership function is associated with the variable short, while trapezoidal functions are used for the two other linguistic variables. The fuzzy logic output is the *DangerLevel* variable, which represents the danger level using three linguistic variables: *low* (between 0 and 0.4), *medium* (between 0.25 and 0.75) and *high* (between 0.6 and 1). These parameters are chosen for a experimental present purpose. With expertise, they can be set to more practical and reliable values.

The fuzzy control inference engine defines the rules that should be applied over the fuzzy values previously obtained from the HeadPose and Distraction variables. More specifically, this engine uses the Mamdani inference method for evaluating the systems rule base and determining the degree of truth of each linguistic variable for the level of danger. The rule base defined for the proposed system is shown in Table 3.6.

In [101], different strategies to combine images from the Internet and simulations are tried and found that the best strategy is to mix images from the Internet and simulations for the training of distraction recognition module. Here, this work tries to mix generated images with images from simulations to train the CNN described in Section 3.2.2 as the distraction recognition module. Images in S_s are captured with a frame rate of 5/sec.

Table 3.6: Fuzzy rules used to calculate the degree of truth of the danger level.

		HeadPose		
		short	**medium**	**long**
Distraction	**short**	low	medium	high
	medium	medium	high	high
	long	high	high	high

To reduce similar images, images of normal driving are resampled to a frame rate of 1.25/sec and resample images of distracted driving to a frame rate of 0.3125/sec. 800 generated images of normal driving, 400 generated images of driving while texting, and 400 generated images of driving while talking on a phone are used. In training dataset, the rate between the number of images of normal driving and the number of images of distracted driving should comply with the rate of normal driving and distracted driving confirmed by Naturalistic Driving Study (NDS) [6]. However, the rate of normal driving and distracted driving activities in our simulation dataset does not follow this percentage, and therefore all datasets are resampled so the rates in both the training dataset and the testing dataset are the same. The testing results are shown in Table 3.7.

The first row in Table 3.7 shows testing resulting on images from simulations of 10 drivers while the model is trained on images of the other 10 drivers. The second row shows results by training the distraction recognition module on images from the Internet and simulations of 10 drivers and testing on images of the other 10 drivers. The third row shows results while training on the mixture of 800 generated images and simulations of 10 drivers. The forth row shows results while training on the mixture of images from the Internet, 800 generated images ,and images of simulations of 10 drivers. All these numbers are the average result of 10 random drivers combinations. These accuracies show that training on the mixture of generated images and images from simulations gives the best performance which is 94.36% and is 2.23% better than the accuracy on the mixture of images from the Internet and simulations. In all these experiments, we stop the training process after the training accuracy achieves 100% and notice that the combination in the second row uses less epochs than the other two. This has a similar effect to early stop

Table 3.7: Testing performance while combining training.

Training Dataset	**Testing Performance On S_s**	
	Accuracy (%)	Cross Entropy
$S_s \times 10$	93.42 ± 3.10	0.21 ± 0.10
S_r & $S_s \times 10$	92.13 ± 1.91	0.21 ± 0.05
S_g & $S_s \times 10$	94.36 ± 2.95	0.18 ± 0.09
S_r & S_g & $S_s \times 10$	91.96 ± 1.65	0.23 ± 0.07

method in network training, and we think this is the reason that the combination in the second row achieves the best performance. Thus, we train the CNN on the mixture of generated images and images of simulations of 20 drivers to form the driver monitoring system and test this system on different distracted driving scenarios of new drivers.

3.4.4 Case Study

Fig. 3.9 shows the danger level for four different scenarios denoted as X-Y, where X represents the driver's direction of view, and Y represents the distraction activity. Fig. 3.10 and Fig. 3.11 illustrate that driver's head movements during these driving scenarios by pitch and yaw angles.

It can be seen from Fig. 3.9 (a) that driving normally while focusing on the frontal view at a low speed (20 km/hour) or a high speed (80 km/hour) makes the danger level stable at 0. In fact, the values of the pitch and yaw associated with this scenario are located below the threshold as shown in Fig. 3.10 (a) and Fig. 3.11 (a), respectively.

The scenario Right-Normal (Fig. 3.9 (b)) means the yaw exceeds the threshold (Fig. 3.11 (a)) and the driver is not using a phone. 7.3 seconds after the driver turns his head to the right, the danger level increases to 1 if he is driving slowly (20 km/hour). Meanwhile, driving with high speed (80 km/hour) while keeping watching the right-side view results in a sharply increasing danger level. In fact, the maximum danger level with 80 km/hour is reached after only 4.7 seconds, compared to 7.3 seconds for 20 km/hour. These tests are

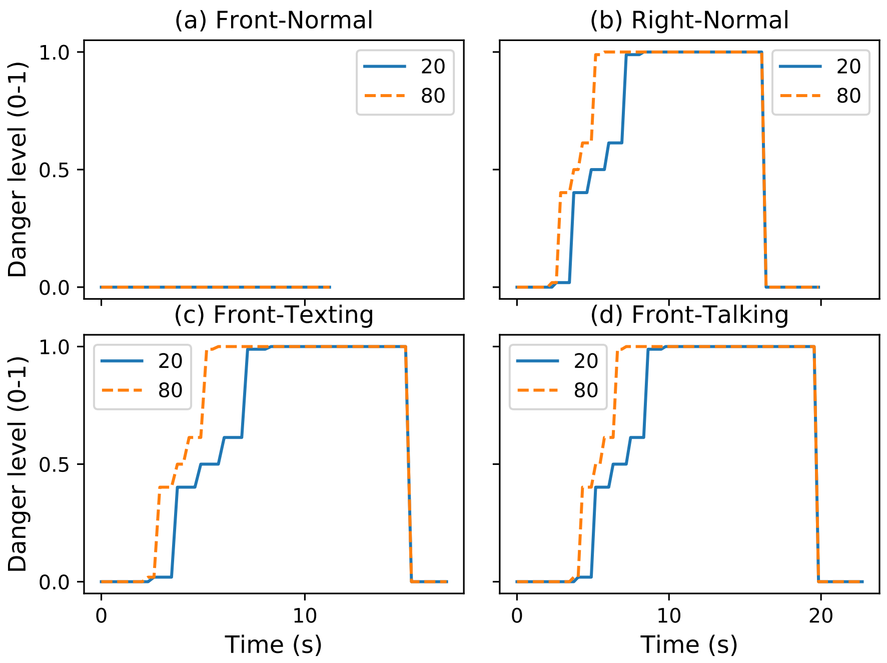


Figure 3.9: Increasing process of danger level. The dash line represents the curve for driving with a high speed (80 km/hour), the solid one is for driving with a low speed (20 km/hour).

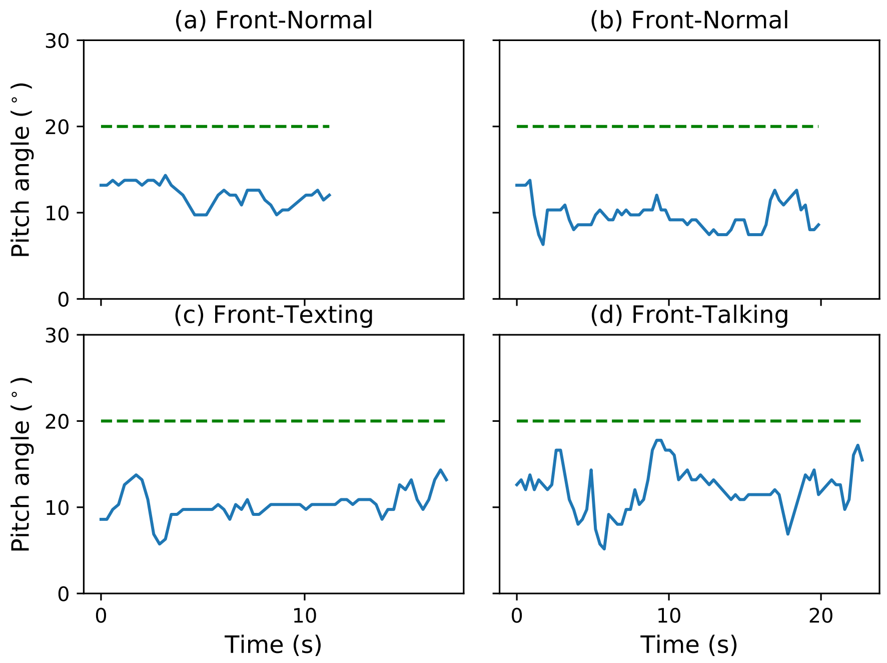


Figure 3.10: Changing processes of the pitch of head pose. The dash line represents the threshold (20°) for deciding whether a driver is focusing on the road, the solid curve is for the pitch angle.

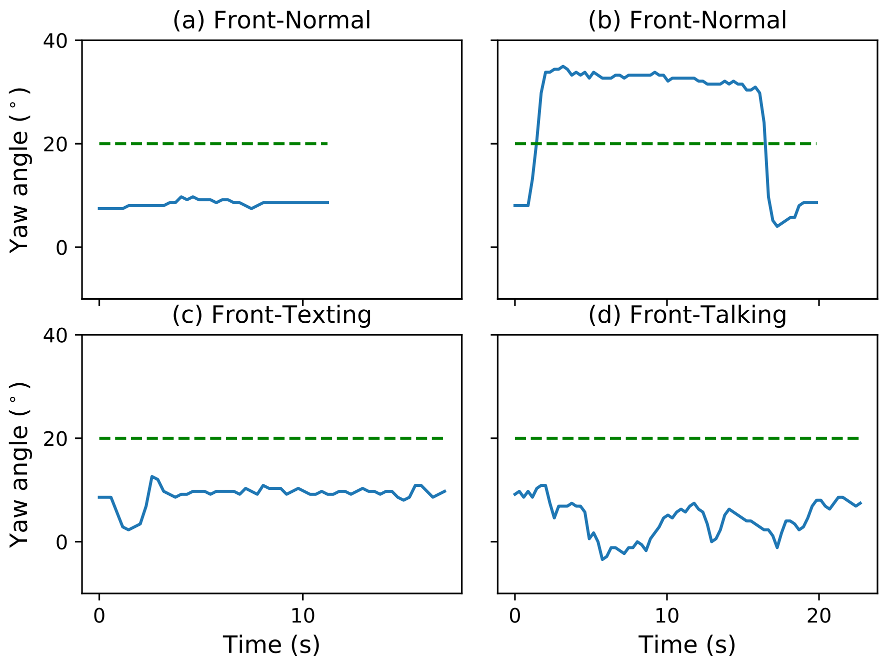


Figure 3.11: Changing processes of the yaw of head pose. The dash line represents the threshold (20°) for deciding whether a driver is focusing on the road, the solid curve is for the yaw angle.

made on a simulator using a specific embedded computer, the results may vary based on the parameters setting (e.g., S in Equation 3.8) and faster processor. But this experiment is only to give insights and not the actual values.

The scenario Front-Texting shown in Fig. 3.9 (c) is considered dangerous even though the driver is holding a phone on a hand but not looking at the phone. Even though the pitch and yaw are below the threshold as shown in Fig. 3.10 (c) and Fig. 3.11 (c), the danger level increases slowly and reaches the maximum danger level 6.6 seconds after the driver picks up his phone. Increasing the speed from 20 km/h to 80 km/h (80-front) makes the danger level reach the maximum level in only 3.7 seconds. The situation Front-Talking shown in Fig. 3.9 (d) is similar to the one in Fig. 3.9 (c), thus the danger level changing processes in them are also similar.

In all distraction driving scenarios (Fig. 3.9 (b)-Fig. 3.9 (d)), the danger level started to increase shortly once a driver is distracted then sharply dropped to 0 as soon as he returned back to normal driving. This experiment shows that our system is able to effectively adjust the danger level output in a short time according to the driver's behavior changes.

The most dangerous situation is illustrated in Fig. 3.12 where a driver is driving while typing on a phone. Notice that the yaw of this driver's head exceeds the threshold since he is watching a phone near his legs. The danger level increases rapidly and reaches its maximum after only 3.2 seconds if he drives with a speed of 80 km/hour, similar results are obtained in the case where he is driving with a speed of 20 km/hour (4.3 seconds).

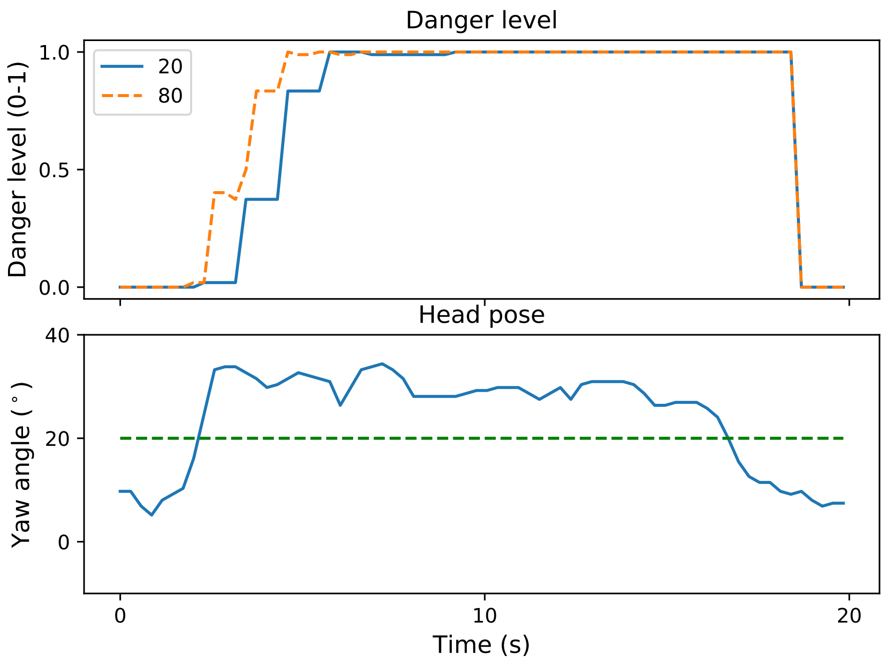


Figure 3.12: Changing processes of the yaw of head pose. The dash line represents the threshold (20°) for deciding whether a driver is focusing on the road, the solid curve is for the yaw angle.

3.5 Summary

This chapter proposes a method to develop end-to-end Convolutional Neural Network-based driver distraction recognition systems that can generalize to diverse driving conditions. The proposed method consists of two stages: developing generative models to produce images of different driving scenarios and developing a discriminative model for image classification. Unlike traditional methods based on image datasets collected by simulation experiments, a diverse dataset of drivers in different driving conditions and activity patterns from the Internet is collected and used to train generative models for multiple driving scenarios. By sampling from these generative models, we augment the collected dataset with new training samples and train a Convolutional Neural Network for distraction recognition. The experiment results demonstrate that the generative models are able to generate images of drivers in different driving scenarios. With augmented images, the DDR system achieves an improvement of 11.45% on image classification performance in a driving simulation environment. Moreover, this chapter presents a novel driver monitoring system, in which a distraction detection system works as a critical information resource. The effectiveness of this monitoring system is also presented by case testing experiments.

Chapter 4

Driver Distraction Recognition Under Various Lighting Conditions

4.1 Introduction

One major drawback for most existing computer vision-based driver distraction recognition methods is that they only work during daytime since they require RGB images as inputs for analysis, including classification, phone detection, etc. As shown in Fig. 4.1, luminance conditions at night are quite different from them in the daytime, and the distribution of RGB images collected at different time of a day varies significantly. Thus, the performance of these recognition systems may degrade in different time of a day. One system that addresses distraction recognition but does not rely on visible light is in [31]. This system works on Near Infrared (NIR) images; after localizing a driver's face in an image, features from image patches around his/her ears are extracted for image classification. Inspired by this work, this work proposes to detect drivers' distraction at nighttime based on in-vehicle NIR images of drivers' body poses.

Another drawback of many existing studies by deep learning is that models employed in them have enormous complexities and sizes, such as VGG-16 in [114], ResNet-50 in [101], and Multiple Scale Faster-RCNN in [38]. The inference processes of these models

(a) An image of cabin environment in the daytime [112].

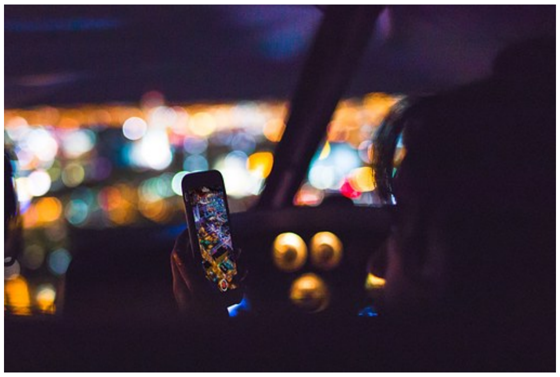

(b) An image of cabin environment at night [113].

Figure 4.1: Comparison of luminance conditions at different time of a day.

involve a large amount of floating-point multiplication-adds computations. Therefore, they require powerful computation devices, e.g., GPUs. Once be deployed in resources-limited environments, the frame rate of inference processes of these models will be low. Thus, to develop real-time recognition systems on resources-limited platforms, one critical step is to design CNNs with small complexities and model sizes.

This chapter explores a driver distraction recognition system that works at both daytime and nighttime by two modes. In each mode, a camera collects corresponding images and sends them to a CNN for classification. A light sensor detects the light condition in the vehicle and enables this system switching between two modes when necessary. Moreover, by a novel network bottleneck, this work proposes a new CNN for the classification of driver images. Three distraction activities are investigated in this work: talking on a cell phone, texting on a cell phone, and interacting with an In-Vehicle Infotainment (IVI) device [115]. The levels of danger by different distraction activities are distinct, and the accurate detection of them enables ADAS providing targeting assistance. The proposed approaches are evaluated on an image dataset of distracted driving collected in a simulator environment with different drivers. Experiment results demonstrate that, even with a small footprint, the proposed system detects driver distraction with the state-of-the-art performance on both modes, 98.87 percentage of images of daytime are classified correctly while 97.80 percentage of images of nighttime are classified correctly. The inference speed of the proposed system is assessed on a resource-limited computing platform, and experiment results show that the proposed network achieves the highest frame rate of inference.

The key contributions are summarised as follows:

- The development of an approach for driving distraction detection under various lighting conditions.
- The design of a novel bottleneck for building CNNs in computer vision applications.
- Comprehensive experiments to collect images of distracted driving under two lighting conditions by 25 drivers and evaluate the proposed methods on this data set.

This chapter is organized as follows. The proposed approach is detailed in Section 4.2. In Section 4.3, the description of the collected dataset, experimental setup, and fine-tuning approaches are given. Analysis of experimental results is in Section 4.4, and finally, Section 4.5 concludes the chapter.

4.2 Proposed Approach

4.2.1 Structure of the Proposed System

To address distraction recognition in both daytime and nighttime, this work proposes using two recognition modes: an RGB mode and an NIR mode. The RGB mode is activated in the daytime and uses images collected from an RGB camera to identify distraction. The NIR mode, on the other hand, is activated in the nighttime and uses an NIR camera. In each mode, the system identifies distraction by a CNN image classifier. To know which mode to activate at any given time, the system uses a light sensor to estimate the in-vehicle luminance conditions. If the luminance is above a certain threshold, the RGB mode is triggered, otherwise, the NIR mode is activated. Fig. 4.2 shows a schematic of the proposed system.

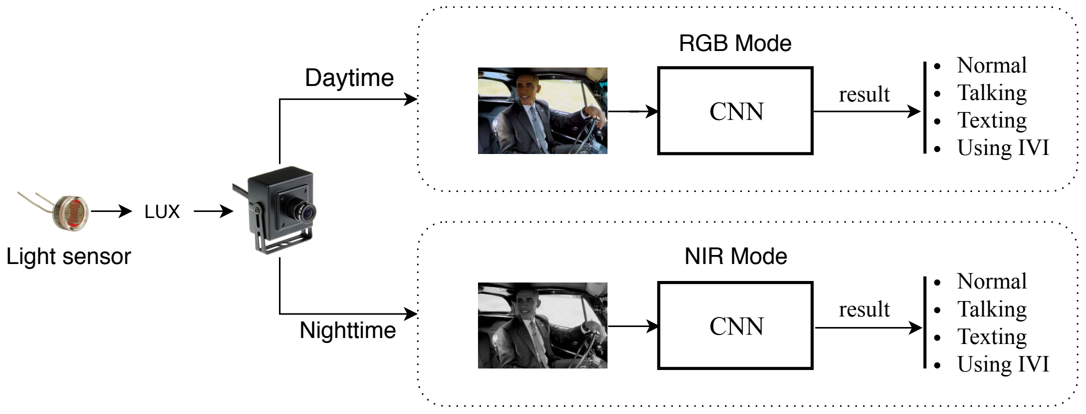


Figure 4.2: A schematic representation of the proposed system.

In each mode, the driver distraction recognition is approached as a four-class image classification problem: one class for "normal driving" and three classes for various distraction activities. The recognition system receives an image I_i at time instance i then classifies it according to the driver body pose in it, and the result is a probability distribution

$$Act_i = [Act_{i1},\ Act_{i2},\ \ldots,\ Act_{i4}], \tag{4.1}$$

where Act_{ij} represents the probability that at time point i the driver is performing a distraction activity j. Moreover, $\sum_{j=1}^{4} Act_{ij} = 1$ since Act_i is a probability distribution. In other words, the classifier models the conditional probability

$$Act_{ij} = P(Act_j|I_i). \tag{4.2}$$

This work employs end-to-end CNN classifiers for image classification. To detect distraction in real-time on a resource-restricted computing platform, the CNNs in this system must be computationally efficient. Most widely-adopted CNN architectures, such as VGG network series [58] and ResNet network series [59], are able to achieve great performances on many computer vision problems, but they require substantial memory sizes and are computation-intensive. To overcome this, this work proposes a new network architecture that is dedicated to resource-restricted environments. The new network is based on a novel bottleneck module of depthwise separable convolutions [116].

4.2.2 Depthwise Separable Convolution

Convolution layers are kernel components of CNNs, and they represent features maps by convolution operations on spatial dimensions. A standard convolution layer convolves every kernel across all features maps and computes dot products between elements of the kernel and an image patch at every position during the forward pass computation. This is illustrated in Fig. 4.3 (a) on the next page, in which the kernel size $k = 2$, the channel number of the input feature maps $D_{In} = 3$, and the channel number of output feature maps $D_{Out} = 3$.

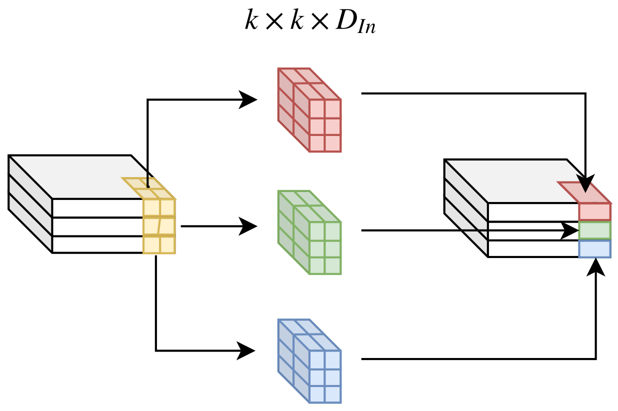
Given input feature maps I of size $H \times W \times D_{In}$ and a standard convolution layer with a kernel size of $k \times k \times D_{In} \times D_{Out}$, the computational cost is

$$H \cdot W \cdot D_{In} \cdot D_{Out} \cdot k^2, \tag{4.3}$$

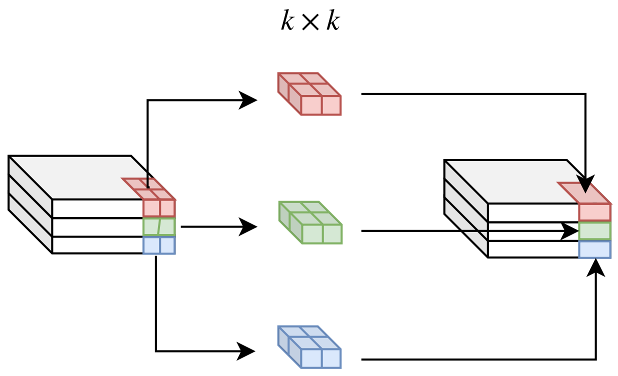
where H and W are the spatial width and height of input feature maps, D_{In} is the depth of input maps (the number of input channels), k is the spatial width and height of a square convolution kernel, and D_{Out} is the depth of output maps. Here, a stride size of 1 is assumed in these convolution operations. The fact that the computational cost is proportional to the spatial sizes of input and output inhibits the usage of a large number of kernels at the beginning phrases of CNNs. Therefore, many CNNs tend to increase the numbers of kernels and reduce the sizes of feature maps synchronously; a good example is VGG16.

While a standard convolution computes dot-products cross channels by a kernel, a Depthwise Separable Convolution layer, originally introduced in [68], computes dot-products for each channel with a corresponding kernel (i.e., depthwise convolutions) then convolutes all channels by kernels of size 1×1 (i.e., pointwise convolutions). This is shown in Fig. 4.3 (b). In a standard convolution layer, non-linearity is added by nonlinear activation after convolutions, while a Depthwise Separable Convolution layer can include two activation operations: one after depthwise convolutions and the other one after pointwise convolutions.

Given inputs and outputs of same sizes as earlier, the computational cost for a Depthwise Separable Convolution layer is the sum of computations in depthwise convolutions



(a) A standard convolution layer with 3 kernels.



(b) A depthwise convolution layer with 3 kernels.

Figure 4.3: Illustration for two different convolution layers.

and pointwise convolutions

$$H \cdot W \cdot D_{In} \cdot D_{Out} + H \cdot W \cdot D_{In} \cdot k^2. \tag{4.4}$$

By replacing a standard convolution layer with a Depthwise Separable Convolution layer, the computation cost is reduced by a rate of

$$\frac{D_{Out} + k^2}{D_{Out} \cdot k^2} \tag{4.5}$$

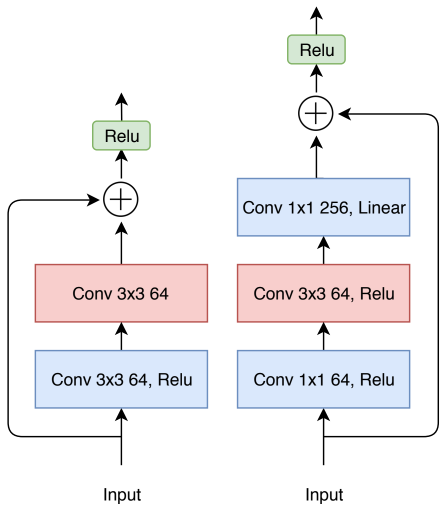
$$= \frac{1}{D_{Out}} + \frac{1}{k^2}. \tag{4.6}$$

For many widely-used CNNs, D_{Out} is significantly larger than k^2, so this rate is dominated by k^2. For a convolution layer with $k = 3$, the computation cost shrinks by a rate of almost $\frac{1}{9}$ and the model size also decreases.

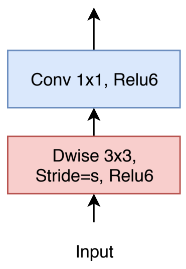
4.2.3 Simplified Inverted Residual Bottleneck

Depthwise Separable Convolution layers contain less trainable parameters, but the naive replacement of standard layers with Depthwise Separable Convolution layers will reduce the representation ability of networks. Therefore, instead of the network architecture that repeats Depthwise Separable Convolution layers as in [64] (shown in Fig. 4.4 (b) on the next page), Sandler *et al.* in [65] proposed a new bottleneck, named as Inverted Residuals Bottleneck. As shown in Fig. 4.4 (c), an input to the bottleneck passes a pointwise convolution layer to increase the dimension of feature maps, then a depthwise layer applies one kernel on each channel, the last pointwise convolution layer restores the dimension of representation to enable an identity shortcut. The increasing and shrinking process of dimension in this bottleneck is opposite to the process in bottlenecks of ResNet, as shown in Fig. 4.4 (a), but the computational cost in this new bottleneck is still significantly reduced.

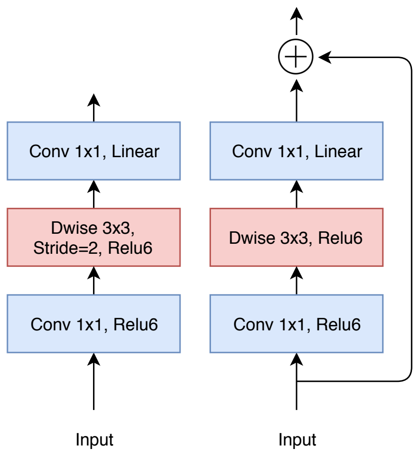
To reduce the computational cost further while adding representation ability to bottlenecks, this work proposes a new bottleneck and names it as Simplified Inverted Residual Bottleneck. As shown in Fig. 4.4 (d), the first layer in this bottleneck is a depthwise layer, which applies multiple kernels on each channel to expend the dimension of features. The



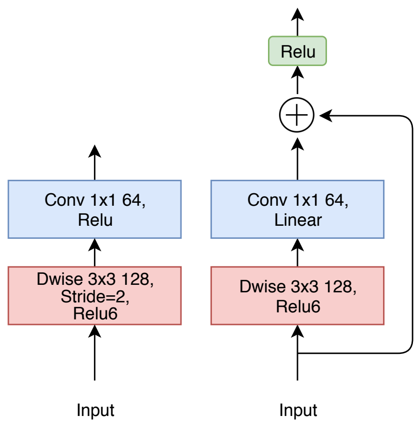
(a) Network blocks of ResNet [59].



(b) Network block of MobileNet [64].



(c) Network blocks of MobileNet V2 [65].



(d) Network blocks of MobileNet S.

Figure 4.4: Architectures of several different network bottlenecks. In these figures, 'Conv 3×3 64' represents a convolutional layer with 64 kernels of size 3×3. Notice that in our bottleneck Relu activation function is used instead of Relu6 after the add operation in the bottleneck body.

second layer performs pointwise convolution to calculate the linear combination of results from the previous layer. If the spatial dimension is not reduced (by a stride of more than 1), a short cut will be inserted into the bottleneck to improve the gradient propagation ability across layers, then the results of add operation will pass a Relu activation. Notice that in our bottleneck Relu activation function is used instead of the Relu6 after the add operation in the bottleneck body.

For an input tensor of size $H \times W \times D_{in}$, the computational cost for an inverted residual bottleneck with a expansion factor of e is

$$H \cdot W \cdot D_{In} \cdot k^2 \cdot e + 2 \cdot H \cdot W \cdot D_{In}^2 \cdot e, \tag{4.7}$$

and the computational cost for our bottleneck with the same expansion factor is

$$H \cdot W \cdot D_{In} \cdot k^2 \cdot e + H \cdot W \cdot D_{In}^2 \cdot e. \tag{4.8}$$

These two numbers show that the proposed Simplified Inverted Residual Bottleneck decreases the computation cost, by $H \cdot W \cdot D_{In}^2 \cdot e$, and the number of training parameters. Through dropping the pointwise convolution layer and moving its functionality to a depthwise layer, the complexity of bottleneck is reduced.

4.2.4 Network Architecture

Besides the design of bottlenecks, there are several hyperparameters in a network architecture that should be set by cross-validation experiments. Due to the limitation of computation resources, this work does not perform grid searching on the network architecture. The main structure of the network in [65] is adopted while the original bottleneck is replaced by the proposed one, and the new network is named as MobileNet S. Obviously, this network structure is not optimal for the proposed bottleneck, but the experiments in the rest of this paper will show that this network can achieve an excellent performance with respect to the accuracy and the inference speed.

Table 4.1 presents the architecture of MobileNet S, in which e is the expansion rate, c is the number of output channels, n is the repeated time of bottleneck, and s is the

Table 4.1: The architecture of MobileNet S.

Input	**Operator**	e	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1×1	-	1280	1	1
$7^2 \times 1280$	avgpool 7×7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1×1	-	1000	-	

size of stride in the first one of repeated bottlenecks. This work does not use the input image resolution and width multiplier as tunable hyper parameters to trade-off between performances and model sizes as limited by our computation capacity.

4.3 Data Collection and Experimental Setup

This section describes the data collection process and presents the experimental setup to evaluate the performance of the proposed approaches.

4.3.1 Data Collection

To evaluate the proposed system, an image dataset of several distraction activities in a driving simulator is collected in this work. A camera is located at the right frontal side of drivers to capture a complete view of drivers' upper body movements. Images were collected on 25 participants of different ages, genders, ethnicities, etc.

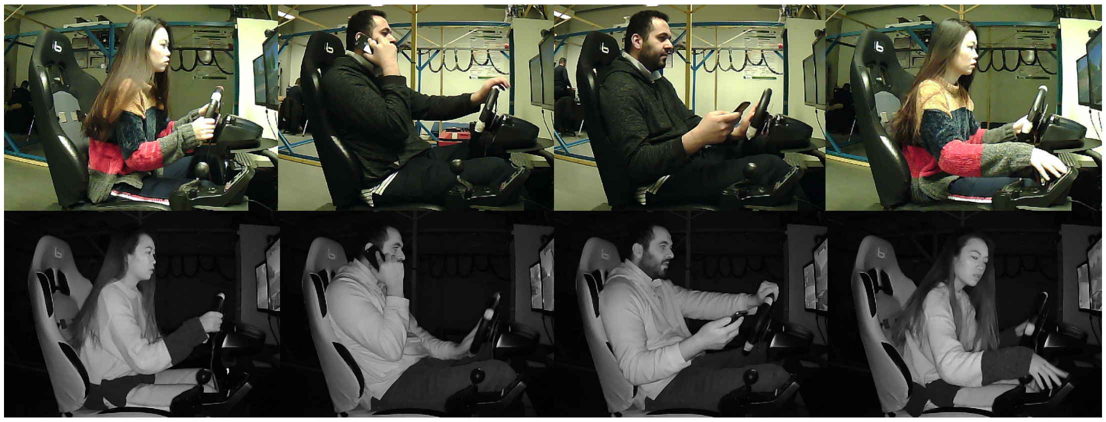


Figure 4.5: Image samples from a simulator environment. The first row shows images taken in the daytime, the second row shows images taken in the nighttime. The first column represents images of normal driving, and the rest three columns represent talking, texting, and using IVI while driving, respectively.

Table 4.2: Number of samples in image datasets.

Class	**Number of samples**	
	Daytime	**Nighttime**
Normal Driving	4993	4893
Talking while driving	9871	9943
Texting while driving	9900	9933
Operating Info while driving	4877	4935

Before experiments, each participant was asked to drive the simulator for a short time to get familiar with the experimental environment. Then, each driver was instructed to perform the following driving activities: normal/safe driving while checking side mirrors occasionally, driving while typing messages with his/her right hand, driving while talking on a cell phone, and driving while operating a device near the gear stick of the simulator to mimic programming an IVI device. Images were extracted from a video stream of 5 frames per second. With this low frame rate, images of adjacent time stamps present different body poses. Fig. 4.5 gives some sample images of different activities. The number of images for each class is given in Table 4.2.

4.3.2 Experimental Settings

The kernel part of the proposed work is to train CNN classifiers for recognition. This work explores and compares several CNNs that were proposed for Imagenet competition [111]. Since a potential future direction is to deploy this system on a small portable computing platform, the proposed network is compared to three existing CNNs (ResNet50, MobileNet, and SqueezeNet). Among them, MobileNet and SqueezeNet own relatively small sizes but powerful representation abilities and ResNet50 was used in our previous research works. The only adjustment made to the original models is the original last fully connected layers are replaced by fully connected layers of 4 nodes.

All models are trained with an Adam optimizer with a learning rate of 1e-3 and a batch size of 32 [106]. The preprocessing transforms of training images include a color jitter with a random brightness factor in the range of $[0.6, 1.4]$, a random contrast factor, saturation factor, and hue factor in the range of $[0.95, 1.05]$. Channel normalization is also used in the images preprocessing. Each model is trained on images of 24 drivers and tested on the other one, these experiments are repeated with 5 different random seeds and report average performance cross 25 drivers and 5 repeats. The performances of all models are measured based on their classification accuracy.

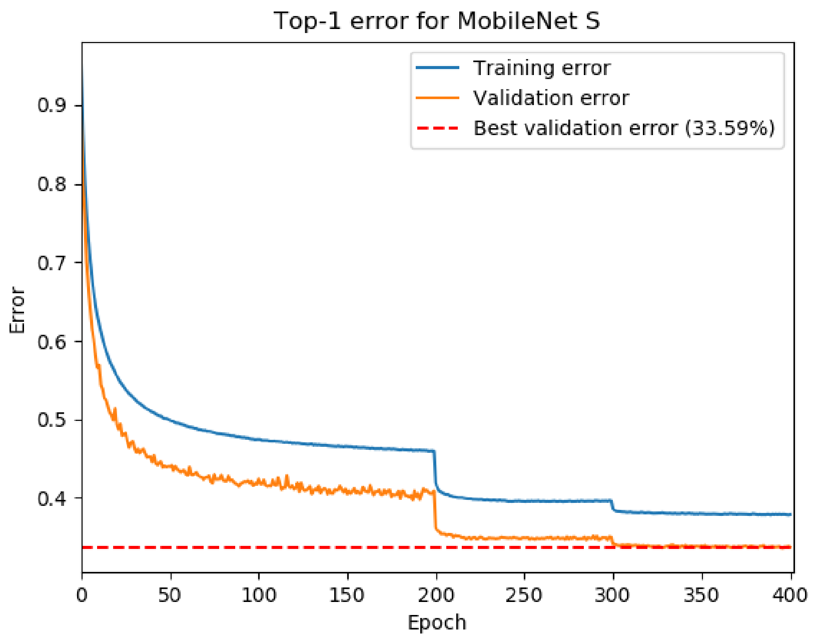


Figure 4.6: The changing process of Top-1 error during the training of MobileNet S. Top-1 error means the class that has the highest prediction probability is used for testing.

4.4 Experimental Results and Analysis

4.4.1 Pretrain Networks on ImageNet

For many computer vision applications, pretraining deep learning models on the ImageNet database provides well placed initial parameters that could benefit subsequent training processes [117]. Following this practice, the proposed network is trained on the training part of ImageNet and evaluated on the validation part. The training process uses a standard SGD optimizer with a momentum set to 0.9, a weight decay rate set to 4e-5, and a batch size of 32. A batch normalization layer is used after every layer, the initial learning rate is 0.01, and the learning rate decays with a gamma of 0.01 after 200 and 300 epochs. Fig. 4.6 shows that the training process is stable after 300 epochs and the validation error does not decrease obviously in the last stage of training.

Table 4.3: Comparison of several network architectures over ImageNet classification error and Complexity. Top-1 and Top-5 accuracy of all models are for their Pytorch implementations [3, 4, 5].

Model	**Complexity**	**Parameters**	**Top-1 Accuracy**	**Top-5 Accuracy**
	(MFLOPs)	(Million)	(%)	(%)
ResNet50 [59]	4087	23.5	76.15	92.87
ShuffleNet V2 [67]	144	2.3	69.40	88.37
MobileNet V2 [65]	299	2.2	72.10	90.48
MobileNet S	170	1.5	66.40	86.86

Table 4.3 compares the model sizes and performances by several selected networks and the proposed one. ResNet50 achieves the best accuracy on both Top-1 and Top-5 tasks but with a price of the most complex architecture, its complexity is 24 times of MobileNet S' and its number of parameters is 14 times more than MobileNet S'. As to two lightweight network examples, MobileNet V2 has a complexity of 299 MFLOPs, and this number is two times of ShuffleNet V2's; meanwhile, these two networks have similar numbers of parameters. Our proposed network has 170 MFLOPs, which is slightly more complex than the smallest one, and it has 1.5 million of parameters, which is the smallest in this table. Due to the not-optimal structure, the proposed network does not give a performance that is comparable to the other two networks. Experiments in the rest of this paper will show that, even with this structure, MobileNet S is still able to achieve an almost identical accuracy and a faster inference speed for distraction recognition to the other two lightweight networks.

4.4.2 Performance in RGB Mode

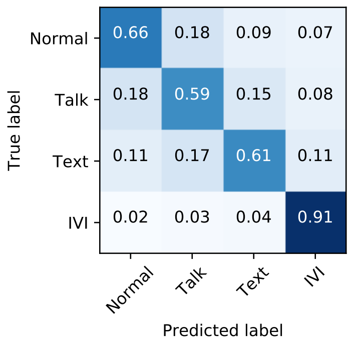
In this section, the performance of the proposed system in RGB mode is evaluated on testing images preprocessed as training images. These testing images are jittered in the same way as training images and these reported results are for images with random bright-

Table 4.4: Comparison of several network architectures over classification performance and Complexity in the RGB mode.

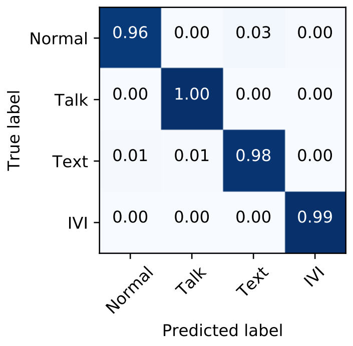
Model	**Multi-class Accuracy (%)**		**Binary-class Accuracy (%)**	
	Non-Pretrained	**Pretrained**	**Non-Pretrained**	**Pretrained**
ShuffleNet V2	66.13±1.86	98.64±0.43	66.16	98.63
MobileNet V2	73.67±1.38	98.87±0.30	73.72	98.88
MobileNet S	73.23±0.76	98.41±0.33	73.33	98.40

ness conditions. Table 4.4 compares the proposed network to two others based on their performances of two classification tasks, complexities, and sizes. In this table, a multi-class accuracy is the accuracy of a task in which each image is classified into one of four classes (Normal, texting while driving, talking while driving, or using IVI while driving.) and a binary-class accuracy means the network is classifying each image into one of two classes (Normal or distracted). Table 4.5 illustrates the testing results on images which are jittered with several fixed brightness factors, while Fig. 4.7 presents the confusion matrices of testing results by these models.

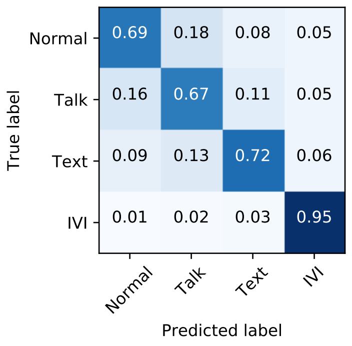
As shown in Table 4.4, pre-trained models achieve better performances for both multi-class and binary-class classifications, and the difference can be as large as 30%. This result justifies this work utilizes models pre-trained on the ImageNet dataset. The 4th column in this table illustrates that ShuffleNet V2, among non pre-trained models, achieves an obviously low accuracy in multi-class classification task while the results of the other two networks are similar. This phenomenon presents that a pre-training process is more critical for the success of adopting ShuffleNet V2 in computer vision applications. The 5th column shows the accuracy of multi-class tasks by pre-trained models. Among the three models, MobileNet V2 achieves the best accuracy, which is 98.87%, while owning the most complex structure. ShuffleNet V2, which has only a half of the complexity and a similar number of parameters of MobileNet V2, achieves a similar accuracy. When comparing the footprints of MobileNet V2 and the proposed MobileNet S, the latter has 57% of the former's complexity and 68% of its number of parameters. Despite this, the difference of accuracy by these two



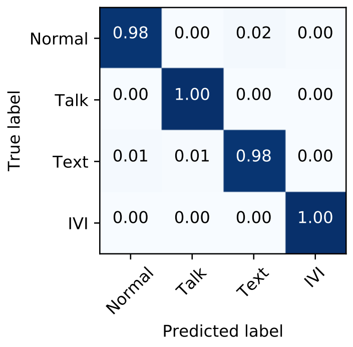
(a) ShuffleNet V2



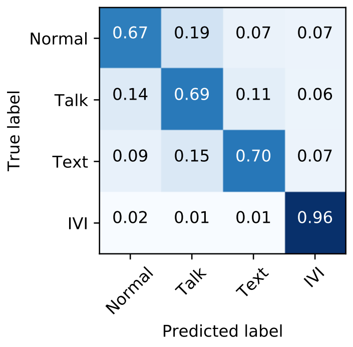
(b) Pretrained ShuffleNet V2



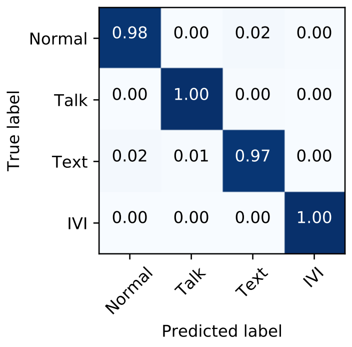
(c) MobileNet V2



(d) Pretrained MobileNet V2



(e) MobileNet S



(f) Pretrained MobileNet S

Figure 4.7: Confusion matrices achieved by different models in day mode. The left fig in each row shows results for a non-pretrained network and the right one is for a corresponding pretrained network.

networks is only 0.46% which is not obvious when compared to the standard deviation of their own accuracy. The 6th and 7th columns are evaluation results based on the same training experiments as the previous two columns while merging three classes of distraction activities into one single class, namely distracted driving. In the new inference process, if one of three distraction classes receives the highest probability, the corresponding image is classified into the distracted driving class. These two columns show a key observation that the performance does not change significantly even the classification task is reduced to a binary one.

Fig. 4.7 illustrates the confusion matrices for testing results of three networks on the multi-class task, in which the diagonal shows the percentages of correctly classified images for the corresponding class. Several key observations from these figures are as follows. First, talking or using IVI while driving is easy to be detected for all networks while misclassification occurs frequently on images of normal driving. Second, images of texting while driving tend to be classified into normal driving or talking while driving. The reason is that drivers' hand positions while texting vary from near their legs to close to steering wheels (similar to normal driving) or even above steering wheels (similar to talking while driving). Third, most misclassified images of normal driving are classified as texting while driving.

In our image dataset, all RGB samples are collected in the same light condition. In the image preprocessing of training, images are jittered with random factors so trained models are able to generalize to images from different lighting conditions. Among several lighting factors, the effect of changing brightness on the performance of distraction recognition systems is the most interested, as it occurs frequently during daily driving. Fig. 4.8 gives image samples that are adjusted with various brightness factors, in which a brightness factor 1.0 means images are not changed. Table 4.5 investigates the robustness of the proposed systems to varying brightness conditions by comparing performances in different brightness conditions by different models. The 1st-3rd rows are results by not pretrained models, and the 4th-6th rows are results by pretrained models. As shown in this table, all networks achieve their best performances when the brightness factor is 1.0. And with this factor MobileNet V2, among all models, gives the best accuracy, which is 98.98%. But for each network, the accuracy only varies marginally with different brightness factors. The standard deviation of MobileNet S's performances with 9 factors is only 0.20%. These results show that the proposed network and training method enable the distraction recognition system to work effectively under different lighting conditions during the daytime.

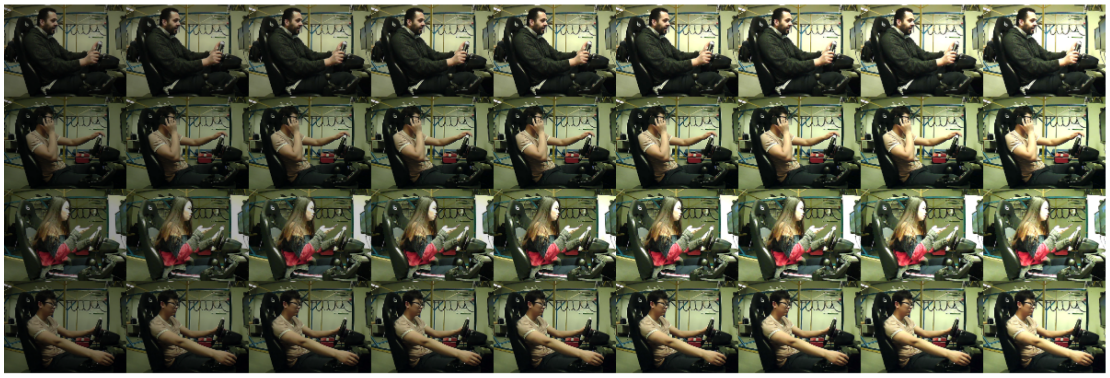


Figure 4.8: Image samples adjusted with various brightness factors. Each row shows a different driving activity. The first column shows images with a brightness factor of 0.6 while the difference between brightness factors of two consecutive columns is 0.1.

Table 4.5: Accuracy (%) by different models in different brightness conditions.

Model	**Brightness Factor**								
	0.6	0.7	0.8	0.9	1.0	1.1	1.2	1.3	1.4
ShuffleNet V2	66.06	67.25	67.55	67.67	66.82	66.20	65.10	63.54	61.50
MobileNet V2	71.62	73.12	73.76	74.53	74.85	74.32	73.76	72.93	71.63
MobileNet S	70.39	71.92	73.07	74.60	74.97	74.65	73.86	72.94	71.47
ShuffleNet V2	98.70	98.74	98.77	98.78	98.76	98.67	98.54	98.39	98.19
MobileNet V2	98.83	98.93	98.96	98.97	98.98	98.91	98.78	98.69	98.59
MobileNet S	98.20	98.37	98.51	98.58	98.60	98.47	98.35	98.23	97.93

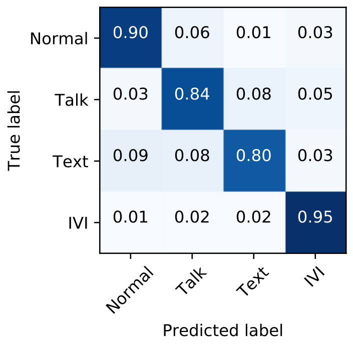
Table 4.6: Comparison of several network architectures over classification performance and complexity in NIR mode.

Model	**Multi-class Accuracy (%)**		**Binary-class Accuracy (%)**	
	Non-Pretrained	**Pretrained**	**Non-Pretrained**	**Pretrained**
ShuffleNet V2	85.49±1.41	98.33±0.16	85.48	98.32
MobileNet V2	92.93±0.30	97.90±0.62	92.90	97.86
MobileNet S	90.44±1.35	97.80±0.37	90.39	97.76

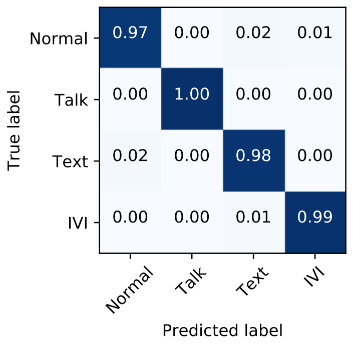
4.4.3 Performance of Networks in NIR Mode

Now this work focuses on the performance of the proposed system in NIR mode. Table 4.6 presents experiment results with different models and on two tasks, and Fig. 4.9 exhibits corresponding confusion matrices. As shown in the 4th and 5th columns, the pre-trained ShuffleNet V2 provides the best accuracy for multi-class classification, which is 98.33%. The accuracy by pre-trained MobileNet V2 is only 0.4% less than the best one, the accuracy by proposed MobileNet S is 0.53% less than it, and the difference between the accuracy of MobileNet V2 and MobileNet S is only 0.1%. Compared to it in the multi-class recognition task, the accuracy by each model does not change significantly in the binary recognition task, as shown in the 6th and 7th columns in Table 4.6. From confusion matrices in Fig. 4.9, one can observe that it is difficult to classify images of normal driving, and many of them are mis-classified as texting or using IVI while driving. Meanwhile, the same as in RGB mode, a part of images of texting while driving are detected as images of normal driving or talking while driving.

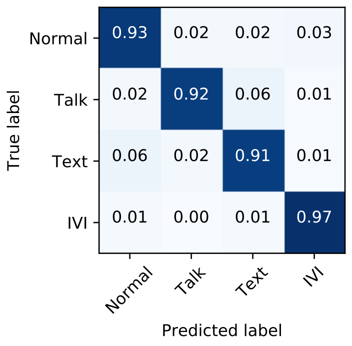
Comparing the performances shown in Table 4.6, Table 4.4, Fig. 4.7, and Fig. 4.9, one can observe that the proposed system achieves a better performance in RGB mode. Furthermore, the accuracy by MobleNet S in the multi-class recognition decreases by 0.6% in the NIR mode, while the one by MobielNet V2 decreases by 0.9%. Even though, the proposed system is able to classify images in various lighting conditions with the state-of-the-art accuracy. The accuracy achieved in our previous work in [101] is 98.30% for RGB images from a fixed lighting condition, while here the accuracy for the same experiment



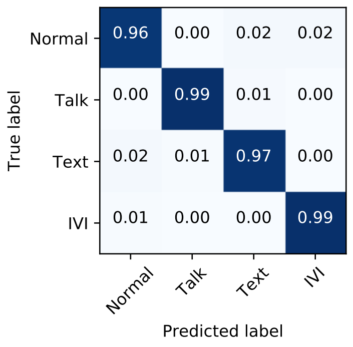
(a) ShuffleNet V2



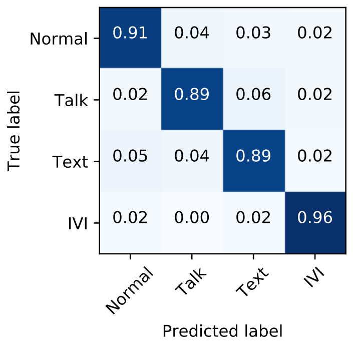
(b) Pretrained ShuffleNet V2



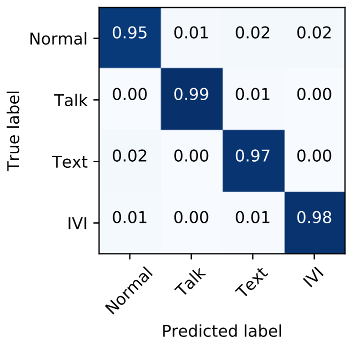
(c) MobileNet V2



(d) Pretrained MobileNet V2



(e) MobileNet S



(f) Pretrained MobileNet S

Figure 4.9: Confusion matrices achieved by different models in night mode. The left fig in each row shows results for a non pre-trained network and the right one is for a corresponding pre-trained network.

Table 4.7: Comparing the inference speeds of networks on two computation platforms by frame rates.

Model	**G1070**	**Nano**
ResNet50	29.81	4.43
ShuffleNet V2	29.81	10.96
MobileNet V2	29.81	12.08
MobileNet S	29.82	**15.85**

is 98.98% by MobileNet V2 and 98.60% by the proposed MobileNet S. The best accuracy from a recent work in [52] is only 91.4% for the binary-class task. Finally, there is still much room for an improvement of correctly classifying driving images. The author conjectures that finding a better network structure for the proposed bottleneck and collecting more training images with different driving environments may help improve the results.

4.4.4 Real-time Inference Speed on a Resource-restricted Computing Platform

The fast inference of distraction recognition enables results to be fused with other information resources for a further situation assessment. Table 4.7 presents the inference speeds of these networks on a desktop computer with an NVIDIA GTX 1070 GPU (G1070) and on an NVIDIA Jetson Nano computer (Nano). Networks investigated in this work are compared to ResNet 50, which was adopted for distraction recognition in our previous research works [101]. The 2nd and 3rd columns in this table show the frame rates of real-time distraction recognition systems, in which networks are required to give a prediction per image from a web-camera. As can be seen, the Frame Per Second (FPS) of inferences of all networks on the desktop computer are nearly the same, but the FPS of inferences on the Jetson Nano are significantly different. The FPS by ResNet50 on Jetson Nano is 4.43 which is not enough for a real-time system. The FPS by ShuffleNet V2 and Mobilenet V2 which have small footprints are 10.96 and 12.08, respectively. The proposed network MobileNet S gives the highest FPS, which is 15.85. These numbers show the proposed network achieves an excellent inference speed, and the author believes this performance improvement arises from the novel bottlenecks proposed in this work.

4.5 Summary

This chapter proposes a novel dual-mode system for driving distraction recognition under various lighting conditions. A novel and efficient CNN bottleneck is introduced to build a network with a small footprint and a high inference speed. The proposed system is trained on images collected under a simulation environment and its performance is evaluated based on its detection accuracy and inference speed. Extensive experiments demonstrate that, with a relatively modest small footprint and design structure, the proposed system achieves the state-of-the-art accuracy on driver distraction detection for both modes, and the proposed network achieves the highest inference speed. The authors believe these enhancements of performance arise from the new network architecture with novel bottlenecks for efficient feature representations.

Chapter 5

Deep Learning-based Driving Maneuver Prediction System

5.1 Introduction

A US statistical study on crashes reveals that improper driving actions account for about 94 percent of crashes in the US [118]. Drivers' decision errors (e.g., misjudgment of time gaps or others' speed) and performance errors (e.g., poor directional control) together comprise 44% of crashes. These facts illustrate the importance of providing immediate assistance to drivers when planning or performing a maneuver [119]. Furthermore, the information about drivers' states and imminent maneuvers enables ADAS to accurately understand driving scenes and provide targeted assistance according to drivers' needs. Hence, next generation ADAS should incorporate not only recognition systems that can perceive and understand driving environments but also recognize and predict driver behaviors. However, the performance of current driving maneuver prediction systems are still far from satisfying OEM requirements to be incorporated as a part of existing ADAS. The best accuracy achieved in recent studies is only around 90% on various data sets, which is insufficient to be implemented in real world applications [19, 88].

In this work, we investigate new methods to improve the performance of driving maneuver prediction. Unlike existing works which handle maneuver prediction under different situations through a single system, our work addresses different driving scenarios by different systems since, for most driving situations, feasible maneuvers do not span all maneuver types; e.g., on a highway, the prediction for a left turn is meaningless. In the proposed system, features of drivers' head movements, vehicle states, and driving surroundings are extracted from driving recordings and fed into a recurrent neural network based prediction model that learns to capture high-level temporal and spatial information from the data. With this information, the system is able to infer drivers' imminent steering actions before they are performed. We evaluate our approach on a driving data set collected under diverse environments with different drivers. Two prediction tasks are employed as examples, i.e., inferring which direction a driver will turn to at a green traffic light and whether he will change his lane while driving. Moreover, this work explores a new driving proficiency testing method which is based on driving maneuver prediction systems.

The rest of this chapter is organized as follows. Section 5.2 details an analysis of drivers' steering procedure in maneuvers, and the proposed prediction method and its alternative architectures. Section 5.3 conducts extensive comparative and ablation experiments to present the validity of the proposed methods. Section 5.4 presents the prototype of bidirectional generalization ability test for driver proficiency assessment and simulation experiments of its core functions. Section 5.5 concludes this chapter.

5.2 Methodology

This section starts with a brief description on how drivers usually perform maneuvers. This description highlights the detailed research scope and some background on the proposed method. The proposed approach is also elaborated in this section.

5.2.1 Driving Maneuvers Analysis

Driving maneuvers follow a three-step model: 1) observation action, 2) decision action, and 3) steering action. An observation action during a maneuver can be divided into

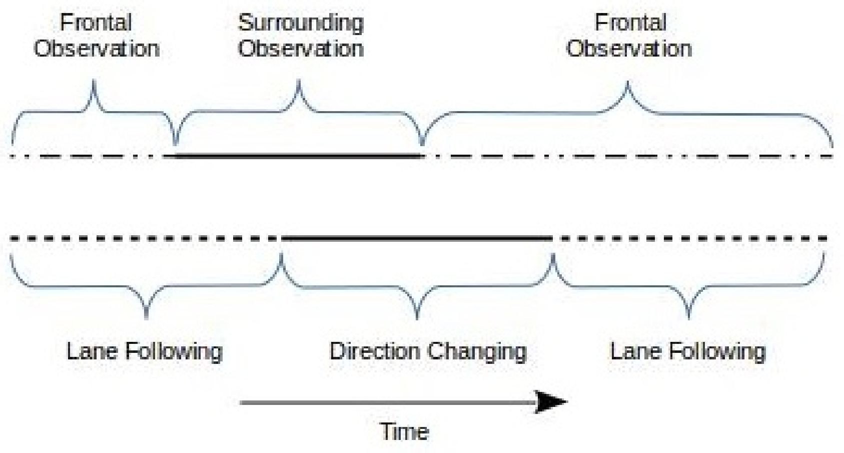


Figure 5.1: Time overlap between an observation action and a steering action.

two stages: a Frontal Observation Period (FOP) and a Surrounding Observation Period (SOP). Driver's observation actions in an FOP aim at following a lane or regulating vehicle position, while observation actions in an SOP aim at deciding whether to take actions or not. The wheel steering action during a maneuver can also be divided into two periods: a Direction Changing Period (DCP) and a Lane Following Period (LFP). In a DCP, a driver steers a vehicle into a new lane. An LFP follows a DCP closely, in which a driver adjusts the position and orientation of his vehicle to the new lane. According to this, a driving maneuver starts with a surrounding observation action and ends with a direction changing action. Before drivers make any steering actions, they observe the surrounding traffic and look for potential confrontation factors. If safety conditions permit, they perform actions. However, in real driving scenarios, time gaps between these steps are not clear, and although a driver may repeatedly check his surrounding, he may start a direction changing action before finishing the last checking action, which means his SOP and DCP overlap. The overlap between an SOP and a DCP is illustrated in Fig. 5.1. This figure shows that the major information used to predict a driver's steering action is gathered from his observation actions. Other preparation actions, such as deceleration, are also performed during this observation period.

Driving maneuver behaviors vary with the color of traffic lights and the layouts of intersections, thus they should be handled separately. Different scenarios will be addressed in the next few paragraphs. In these driving maneuvers, even without obvious violations of traffic rules, driving risk persists due to inaccurate estimations of time windows and failures to detect objects in blind spots. ADAS with abilities to predict the next maneuver and provide corresponding assistance should be able to reduce the driving risk.

For an intersection controlled by traffic lights, if there are three lanes on the road, a common layout is: the left lane can be used for a left turn, and the center lane for going straight, and the right lane for a right turn. In some intersections with three lanes, the middle lanes can also be used for the right turn. At an intersection with two lanes, a driver can turn left only in the left lane and turn right in the right lane, but he may go straight in both lanes.

Turning maneuvers on red traffic lights are strictly controlled by traffic rules: left turn is forbidden, and drivers must stop and wait for a green light; to turn right, a driver must stop and wait for a safe time window. A possible risk of turning right at a red light is overestimating the distance between the ego-vehicle and the vehicle coming from the left direction which may lead to a rear-end collision.

Maneuvers on green lights are allowed by traffic rules, and drivers do not have to stop to check before changing directions, which makes these maneuvers risky. A left turn on a green traffic light is risky when a driver overestimates the time window before the vehicle in the opposite direction arrives at the intersection. Right turns on green lights could be dangerous to bicyclists on the right side and pedestrians on the zebra crossing. Before reaching an intersection with a green light, a driver preparing for a maneuver would control the speed of the vehicle within an appropriate range and start an observation action. Thus a deceleration action and observation action are strong hints as to whether this driver will go straight or make a turn.

Lane changes are another kind of common driving maneuver in which drivers must keep enough distance from vehicles when changing lanes. Experienced drivers check side mirrors and blind spots before taking actions. The acceleration action before lane changing is also a common preparation action. During a lane change, an accident may happen if the vehicle

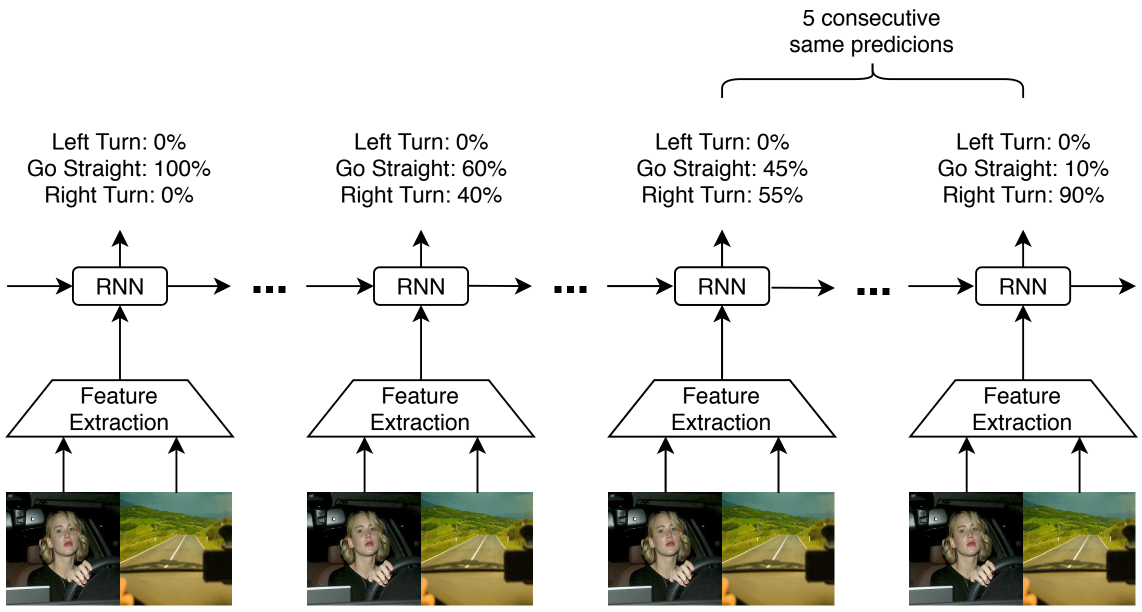


Figure 5.2: Structure of the proposed driving maneuver prediction network.

in the blind spot is ignored or the vehicle behind the ego vehicle is approaching at a higher speed.

5.2.2 The Structure of Driving Maneuver Prediction Systems

Based on the analysis in the previous section, we propose to infer drivers' direction changing actions based on information about their observation actions, vehicle states, and road information. To be specific, we develop a system to predict whether a driver will go straight or turn at an intersection with a green light and whether a driver will change his lane during driving. Even through there are many other driving situations in which maneuver prediction would be helpful for ADAS, we focus on these two since they occur frequently during driving. By restricting the input information to drivers' observation actions between the beginning of SOP and the beginning of DCP, the prediction system can predict drivers' next maneuvers before they take steering actions.

Since drivers' observation actions can be represented by videos of their head movements, we propose to use these videos as one input resource for our prediction system. The second input resource is vehicle speed since several previous works suggest that the temporal information in a speed sequence could represent a deceleration or acceleration action and is important for the prediction [79]. The third input resource is the number of lanes beside the discussed vehicle.

An overview of the proposed driving maneuver prediction system is presented in Fig. 5.2. In this prediction system, which is a variant of RNN, the Feature Extraction module takes a video of a driver's face as input and extracts high-level visual features at each time t. These visual features are concatenated with the speed value and the number of lanes as inputs to an RNN which exploits temporal information among these inputs at different time instances to infer possible future maneuvers. The number of lanes on each side of the ego vehicle should be extracted by a lane detection system, which is not the focus of this work, so we labeled this feature manually during database labeling. Concisely, the input to our system is an information sequence

$$[x_{t-T},\ x_{t-T+1}, \cdots,\ x_t], \tag{5.1}$$

where x_t represents the feature vector of a driver's face, vehicle speed, and the number of lanes at time instance t. The output is a probability distribution sequence

$$[m_{t-T},\ m_{t-T+1}, \cdots,\ m_t], \tag{5.2}$$

where

$$m_t = [m_t^1,\ m_t^2, \cdots,\ m_t^k], \tag{5.3}$$

and m_t^k represents the probability that a driver will take a maneuver action k in the near future. Hence, at any time t, the maneuver prediction system models the conditional probability

$$m_t^j = p(m = j | [x_0, \cdots,\ x_t], \Theta),\ j \in \{1, \cdots,\ k\}, \tag{5.4}$$

where $[x_0, \ldots,\ x_t]$ represents a feature sequence and it includes input x_t and inputs at previous time instances, and Θ represents all parameters in the model. This recurrent neural network may not memorize information from all previous time instances, so in this equation $[x_0, \ldots, x_t]$ just represents that previous input information is used to infer future maneuvers.

The RNN in this system is trained in a sequence-to-sequence manner. Every input sequence is labeled with the maneuver it represents by a label sequence the same length as the input. For each input sequence, this system will output a prediction for each feature frame as an output sequence. The training process will compare the label sequence to this output sequence to evaluate a loss and optimize the RNN using a variant of the stochastic gradient descent method. After this training procedure, the system will learn to infer drivers' next maneuvers based only on partial information.

5.2.3 Recurrent Neural Networks

Recurrent Neural Networks are designed to process temporal data sequences and exploit information dependency at different time instances to make accurate predictions.

Given an input sequence of observation

$$X = [x_{t-T},\ x_{t-T+1}, \cdots,\ x_t],$$

where the length of the sequence is $T+1$ and $x_t \in \mathbb{R}^n$, an RNN generates a sequence of hidden variables

$$H_t = [h_{t-T},\ h_{t-T+1}, \cdots,\ h_t]$$

via a non-linear cell. The hidden variable will store information from the current and previous inputs. By adding a fully connected layer after the hidden layer, the network produces a sequence for the last classification or regression results.

The function for hidden variables is

$$h_t = \hat{h}(W_{xh}x_t + W_{hh}h_{t-1} + b_h), \tag{5.5}$$

and the function for output variables is

$$o_t = f(W_{ho}h_t + b_o), \tag{5.6}$$

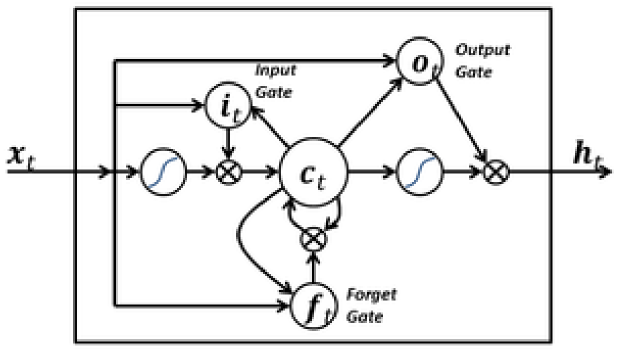


Figure 5.3: The structure of an LSTM node. Image courtesy of [2].

where $\hat{h}$ and f are two nonlinear activation functions. W_{xh}, W_{hh}, and W_{ho} are transfer parameters, while b_h and b_o are bias parameters. All these parameters are trainable. As the equations show, the hidden variable h_t at time t is related to not only the input x_t at time t but also the hidden variable at the previous time instance $t-1$. In this way, the network can infer the current result based on the current and previous input information.

Long Short-term Memory

The training process of RNN suffers from a phenomenon known as exploding and vanishing gradient. Long Short-term Memory is designed to avoid these problems and achieve long-term memory [2]. The architecture of an LSTM node is shown in Fig. 5.3. There is an input gate, an output gate, a forget gate, and a hidden memory cell in each LSTM node.

The hidden memory cell C_t in LSTM stores information from previous inputs. The input gate

$$i_t = \sigma(W_i \cdot [h_{t-i}, x_t] + b_i) \tag{5.7}$$

decides which input information from

$$\bar{C}_t = \tanh(W_C \cdot [h_{t-i}, x_t] + b_c) \tag{5.8}$$

should be updated to the hidden memory cells. In these equations, W with a subscript represents a corresponding transfer matrix and b with a subscript is a base matrix. h_t

represents the hidden states which will be given later, and σ is the sigmoid activation function.

A forget gate

$$f_t = \sigma(W_f \cdot [h_{t-i}, x_t] + b_f) \tag{5.9}$$

decides to remember or drop the information from the previous hidden memory cells in each update. Thus, the update to the hidden memory cell will be

$$C_t = f_t * C_{t-1} + i_t * \bar{C}_t, \tag{5.10}$$

where $*$ means the element-wise production.

The hidden state h_t is related to the hidden memory cell C_t, the hidden state at the previous time point h_{t-1}, and the input x_t at time point t as

$$o_t = \sigma(W_o \cdot [h_{t-i}, x_t] + b_o) \tag{5.11}$$

$$h_t = o_t * \tanh(C_t). \tag{5.12}$$

It is also possible to connect a fully connected layer with the hidden node to obtain the final prediction output y_t. Then we have the final output

$$y_t = W_y h_t + b_y, \tag{5.13}$$

where W_y and b_y are trainable parameters in a fully connected layer.

Bidirectional Recurrent Neural Networks

In basic RNNs, recurrent operations start from one end of an input sequence and stop at the other end. Thus, at each time step, only input information at or before that time will be used for a prediction inference. Bidirectional Recurrent Neural Networks overcome this with parallel consecutive recurrent operations from two directions [120], as illustrated in Fig. 5.4. This structure is critical for many sequence processing tasks in which whole feature sequences are available for predictions [121, 122].

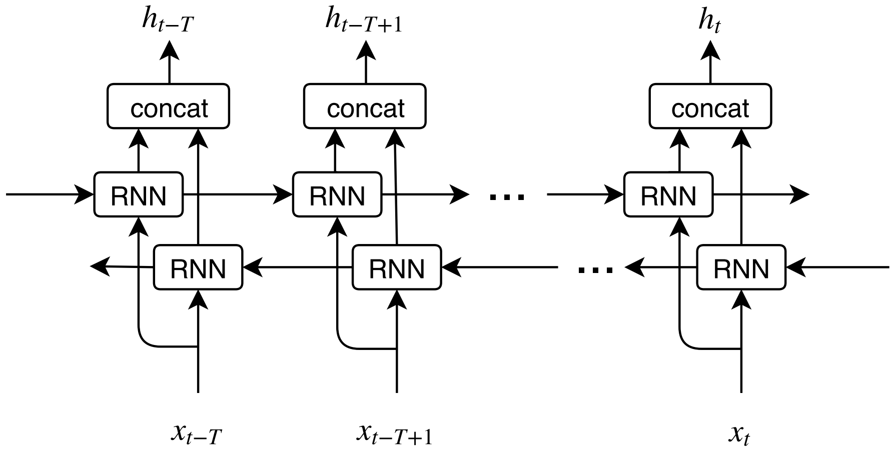


Figure 5.4: The structure of a Bidirectional Recurrent Neural Network.

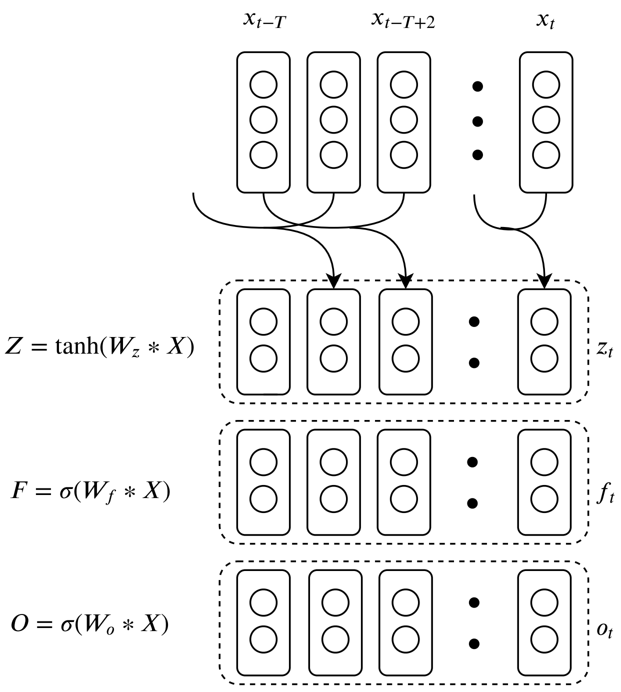
Quasi-recurrent Neural Networks

Temporal dependency is achieved by delivering hidden states h_t to operations at the next timestep in RNNs, while spatial dependency is achieved by layer-wise convolution operations in Convolutional Neural Networks. Quasi-recurrent Neural Networks (QRNNs) combine the advantages of these two kinds of operations for sequence data processing. The first layer in a QRNN is a convolutional layer on the timestep dimension, the following layer is for hidden states propagation [123]. QRNNs could reveal temporal relationships between adjoining feature steps which could be vital for many sequence processing tasks [124, 125].

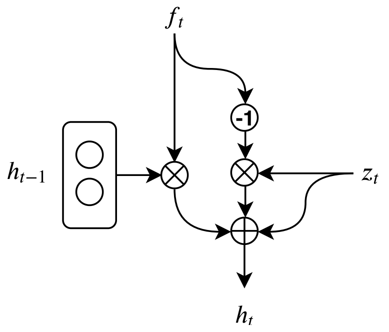
Given an input sequence $X = [x_{t-T},\ x_{t-T+1}, \cdots,\ x_t]$, a convolutional layer with m filters performs convolution operations on the timestep dimension. The convolution operation at any timestep should not use future features to infer current intermediate states. Thus, a filter of width k convolute only feature x_{t-k+1} to feature x_t to calculate z_t, as shown in 5.5 (a),

$$Z = \tanh(W_z \odot X), \tag{5.14}$$

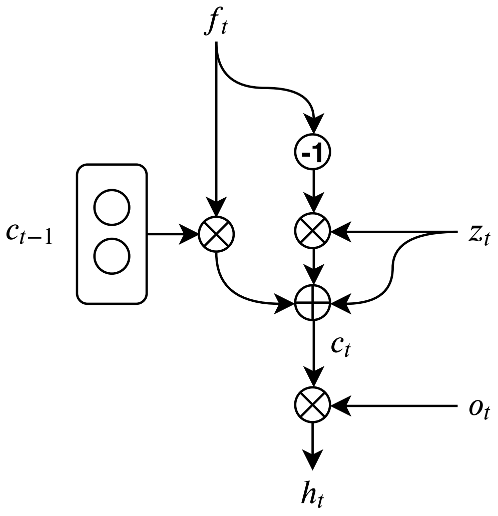
where $W_z \in \mathbb{R}^{k\times n\times m}$ are the convolutional filter weights and $\odot$ denotes the convolution operation.



(a) Convolutional component in a QRNN cell. The filter width is 3, and x_{t-T}, x_{t-T+1}, and x_{t-T+2} are used to calculate z_{t-T+2}, f_{t-T+2}, and o_{t-T+2}.



(b) f-pooling in a QRNN cell.



(c) fo-pooling in a QRNN cell.

Figure 5.5: The structure of a QRNN node.

Another two banks of convolutional operations are performed followed by element-wise nonlinear activation functions to calculate the forget gate f_t and the output gate o_t,

$$\begin{aligned} F &= \sigma(W_f \odot X), \\ O &= \sigma(W_o \odot X), \end{aligned} \tag{5.15}$$

where W_f and $W_o \in \mathbb{R}^{k \times n \times m}$ are the convolutional filter weights.

Similar to operations in a traditional LSTM cell [2], the hidden state h_t is calculated as

$$h_t = f_t * h_{t-1} + (1 - f_t) * z_t, \tag{5.16}$$

where $*$ denotes element-wise multiplications, as shown in 5.5 (b).

Alternatively, the calculation of hidden states may also involve an output gate, as shown in 5.5 (c),

$$\begin{aligned} c_t &= f_t * c_{t-1} + (1 - f_t) * z_t, \\ h_t &= o_t * c_t. \end{aligned} \tag{5.17}$$

We can connect a fully connected layer to the hidden layer to obtain the final prediction output y_t. Then, the final output can be represented as

$$y_t = W_y h_t + b_y,$$

where W_y and b_y are trainable parameters in a fully connected layer.

5.3 Experiments and Evaluation

In this section, we first provide an overview of our dataset and experimental protocol. Then, we present and analyze the quantitative results and evaluations.

5.3.1 Database Collection and Annotation

To train the proposed prediction system, we need a driving maneuver database. A related database is introduced in [89], which includes 700 maneuvers containing 274 lane changes, 131 turns, and 295 randomly sampled instances of going straight. Several cars were used for data collection in driving experiments, so the camera positions vary among samples. However, the authors annotated the starting times of cars touching lane marks (for lane change maneuvers) or cars starting to yaw (for turn maneuvers) and the ending times of steering actions, but did not annotate the beginning and ending times of observation actions or starting times of steering actions [89]. For many samples, the beginning parts of observation actions are ignored, while these parts are critical to training a system that makes early predictions. Moreover, left (and right) turns before different traffic lights are treated as one kind of maneuver, while as we discussed previously these maneuvers are quite different and should be handled accordingly.

Given these facts, we collected and annotated a new database for our research work here. Only one vehicle is used in our experiments, so the camera position and background of those images do not change significantly across videos. Our database does not include actions at red traffic lights.

A total of 7 subjects (4 males and 3 females) who have valid Canadian driving licenses participated in the data collection experiments. Each driver was required to make 20 left turns, 20 right turns, 20 right lane changes, and 20 left lane changes. We did not give drivers any instructions about how to perform maneuvers, so they drove under almost naturalistic conditions in the city of Waterloo, Canada. Videos in our database last for 7.87 hours with a frame rate of 30 frames per second. The database includes diverse weather conditions and landscapes. In our driving experiment, we used a dual-lens dashcam to record both inside and outside views of the car, and OBD II software to write the speed of the vehicle to a comma-separated values (CSV) file.

Our driving maneuver database is annotated manually. The starting and ending time stamps for both SOP and DCP are annotated by checking videos frame-by-frame. For every maneuver, the traffic sign and the numbers of lanes on the right side and left side of the ego-vehicle are also annotated. The annotation method is described as follows:

1) For a lane change maneuver, we labeled the starting time of SOP as the time that the driver started to turn his head to check the rear mirror, the blind spot, or a side mirror, and we labeled the ending time of SOP as the time when his head returned to focus on the

frontal view. The starting time of a DCP was labeled as the time when the vehicle started to approach a lane mark, and the ending time point was the moment that the orientation of the vehicle was no longer changing.

2) For a right or left turn at a green light, we labeled the starting time of DCP with the time that the vehicle reached the ending line of the lane, and labeled the starting point of SOP with the time step of 2 seconds before the starting time of DCP. The ending time of DCP, the number of lanes on the left side and right side of the ego-vehicle were also labeled.

3) During lane change maneuvers, drivers do not always check mirrors and blind spots. We labeled the orders of their observation actions during maneuvers to understand how drivers perform these observation actions. Take a left lane change as an example; if the driver checks the rear mirror first, then the side mirror, and the blind spot last, the observation actions in this maneuver are labeled as "RSB", which represents the rear mirror, the side mirror, and the blind spot respectively.

Fig. 5.6 presents the distribution of time lags between starting observation actions and starting steering actions of all drivers for lane change maneuvers. It shows that most time lags are between 1 to 3 seconds, and the average time lag is 2.03 seconds. The predicted average Time to Steering Acts for testing results should be less than or close to this number.

The numbers of maneuver samples in our database are shown in Table 5.1. To train our network to predict left and right lane changes, negative samples are those of lane following which are selected randomly from driving records in which drivers are following lanes.

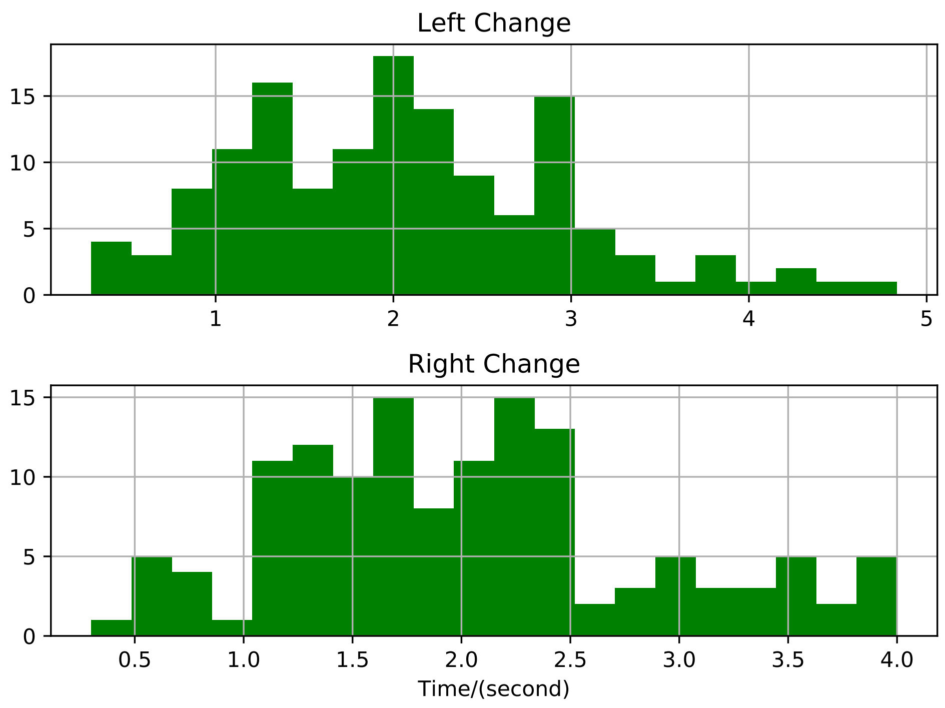


Figure 5.6: Histogram of time lags between starting observation actions and starting steering actions.

Table 5.1: Number of maneuver examples.

Maneuver	Traffic sign	Number
Right turn	Green	61
Left turn	Green	45
Go straight	Green	129
Right lane change		138
Left lane change		142

5.3.2 Feature Extraction and Evaluation Protocol

From our raw data sequences, we extract multi-modality feature sequences as in [87, 88]. The feature of whether a vehicle is within 15 meters of a road artifact (e.g., intersections, highway exits) is not included in our database due to the limitations of experimental equipment.

Algorithm 1 Testing algorithm

```
Initialize: $k \leftarrow 0$, Maneuver $\leftarrow 1$
Input: Feature sequence $[x_0,\ x_1, \cdots,\ x_T]$
Output: Next maneuver Maneuver
  for $t$ is 0 to $T$ do
      Estimate probability $m_t^j$ for each maneuver $j$
      Prediction $= \arg\max_{j \in \{1,\cdots,k\}} m_t^j$
      if Maneuver is Prediction then
          $k \leftarrow k + 1$
          if $k$ is 5 then
              break
          end if
      else
          Maneuver $\leftarrow$ Prediction, $k \leftarrow 0$
      end if
  end for
  return Maneuver
```

To demonstrate the capabilities of the proposed system, we evaluate it for lane change prediction (LCP) and green light turn prediction (GLTP) tasks and compare it with the state-of-the-art work. All systems are evaluated based on their prediction accuracy and prediction times to steering actions (TTS), i.e., the time between making a correct prediction and when a driver starts steering actions for a given maneuver. Algorithm 1 shows the testing steps for maneuver prediction systems. While passing a feature sequence (of a frequency of 5/second) into a model, it assigns a probability for each possible maneuver at every time step. At each time step, the maneuver with the highest probability is the predicted one. The system makes an effective prediction only if the same maneuver is predicted at 5 consecutive time steps. For an input sequence, a prediction is correct if that maneuver is performed seconds later. We calculate the average percentage of maneuvers classified correctly and label it as **AccAction**. Only for correct lane change or turn maneuver predictions, the intervals between the time of prediction and the time that a driver starts steering actions are meaningful and used to calculate the **Average TTS**.

Hyper parameters in our models are chosen by cross-validation on data from random drivers. For each system, we train it on data from 6 drivers and test it on the left out driver. We repeat this process for each driver and execute code with different random seeds 5 times. All results in the rest of this chapter are the average values of these experiments.

5.3.3 The Information on Observation Actions Improves Maneuver Prediction

Drivers' observation actions in surrounding observation periods aim at deciding whether to take actions. To present the importance of the information on these observation and preparation actions to the performance of maneuver prediction systems, we conducted a series of experiments with databases in which different parts of feature sequences were removed.

We first built the original database of lane change maneuvers by mapping each lane change maneuver into a feature sequence between the beginning and ending of SOP. This original database contains complete information of drivers' observation and preparation actions before lane changes. Then, several training databases are formed by removing different numbers of feature frames from the beginnings of sequences; the testing database is left unsullied. We train the same LSTM-based prediction system on these different training databases and test them using the same method. The evaluation results by classification performance and average TTS are presented in Table 5.2 and Fig. 5.7. We stop at removing 7 frames since a part of training samples will be removed if more than 7 frames are removed. The last row in this table shows testing results of models that are trained on the database of feature sequences of driving records between the beginnings and the endings of DCP.

Table 5.2: Prediction performance while removing different numbers of feature frames.

Preframe	AccAction (%)	Average TTS (seconds)
0	89.59 ± 0.41	1.56 ± 0.06
1	88.41 ± 0.91	1.50 ± 0.04
2	86.45 ± 0.96	1.38 ± 0.07
3	85.90 ± 0.86	1.41 ± 0.03
4	85.23 ± 0.99	1.38 ± 0.03
5	82.59 ± 1.80	1.37 ± 0.03
6	84.14 ± 1.39	1.41 ± 0.03
7	82.53 ± 1.31	1.37 ± 0.04
Act	67.96 ± 1.04	1.01 ± 0.10

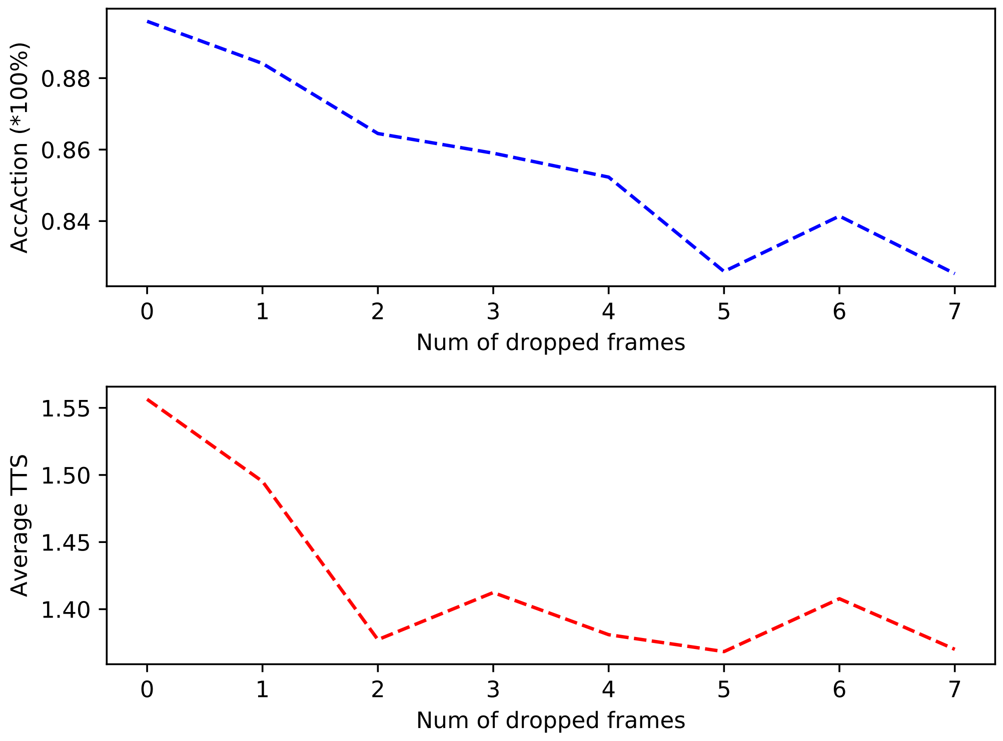


Figure 5.7: Comparison of performance by models trained on different datasets.

As shown in Fig. 5.7, there is an obvious trend that, as more information of observation actions is removed from the training database, both the prediction accuracy and average TTS drop significantly. By training on the original database, the achieved testing accuracy is 89.59% and average TTS is 1.56 seconds. Removing the information of the 1st frame induces a decrement of 1% in the testing accuracy while the reduction of average TTS is trivial. The accuracy and average TTS keep decreasing as we remove more information regarding observation actions from the training database. After removing 7 frames from training samples, the testing accuracy drops to 82.53% which is 6.94% less than the accuracy achieved by training on the original database, while the average TTS drops to 1.37 seconds which is 0.2 seconds less. The last row in the table shows that models trained on feature sequences between the beginning and end of DCP give poor performance on this maneuver prediction problem, which is 21.51% less than the accuracy and 0.56 seconds less than the average TTS achieved by training on the original database.

5.3.4 Models to Better Capture Temporal and Spatial Information

Besides correctly labeled training data, a model that is able to draw and fuse historical information effectively from sequences is also essential for accurate maneuver prediction. We propose using a QRNN to learn the correlation between feature sequences and driver maneuvers for a maneuver prediction system. We replace QRNN with other sequence-to-sequence RNNs to baseline with two state-of-the-art systems, one based on LSTM [87] and the other based on bidirectional RNN [88].

The configuration of LSTM follows conclusions in [87] and [88] in which every hidden cell includes 64 nodes and there is only one layer of LSTM cell. The configuration of BDRNN follows conclusions in [88], in which a unidirectional GRU cell of 128 nodes atop a bidirectional LSTM cell of 64 nodes. As in [88], the implementation of the bidirectional model employs a slide window method to make real-time predictions. The system of QRNN contains 2 layers of QRNN cells and each cell contains 64 nodes. Another critical hyper-parameter of QRNN is the size of convolutional windows; in the LCP task this number is set to 2 while in the GLTP task it is set to 1 by cross-validation experiments. An Adam

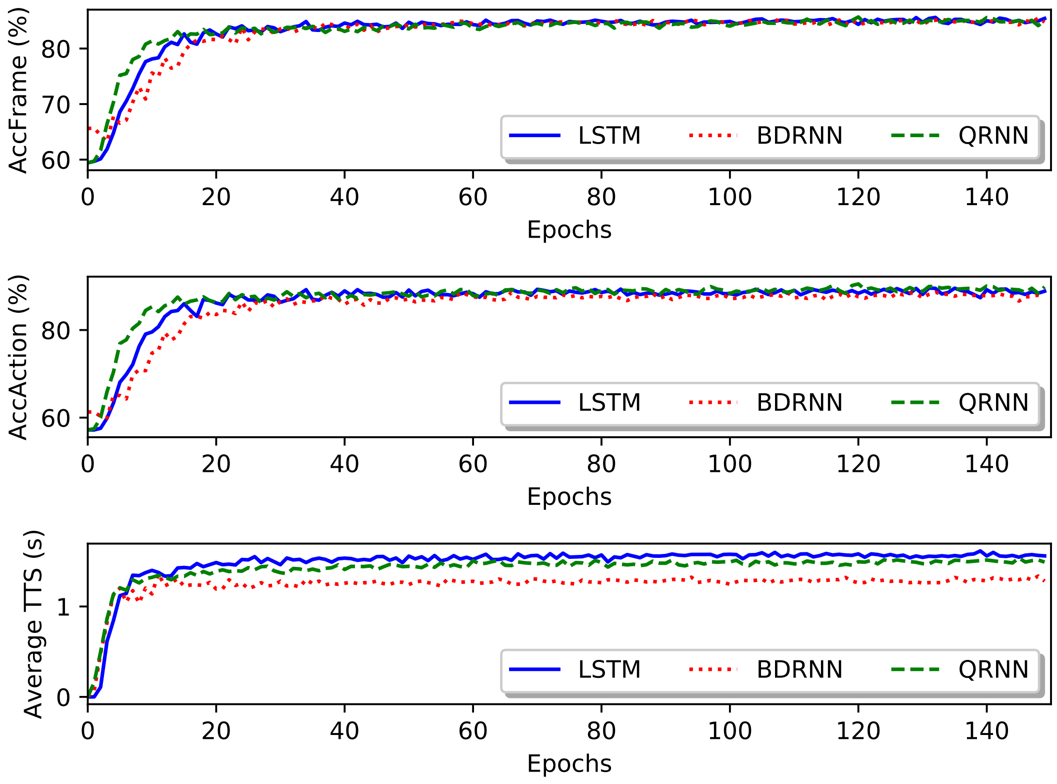


Figure 5.8: Testing performance of lane change prediction by different models with different training epochs.

optimizer with a learning rate of 0.001 and a batch size of 6 is adopted in all training processes [106].

Fig. 5.8 presents the testing performance of the proposed lane change prediction system and baseline models during the training process, and Fig. 5.9 presents green light turn prediction task performance. The AccFrame in both figures represents the percentage of frames that are assigned with correct labels. According to these two figures, all three metrics (AccFrame, AccAction, and Average TTS) grow rapidly in the first 20 epochs, then the growth tends to slow down and become tardy. Curves in Fig. 5.8 are smooth after 40 epochs of training, while curves in Fig. 5.9 keep oscillating due to a relatively small number of training samples. All training processes are stopped at 150 epochs since

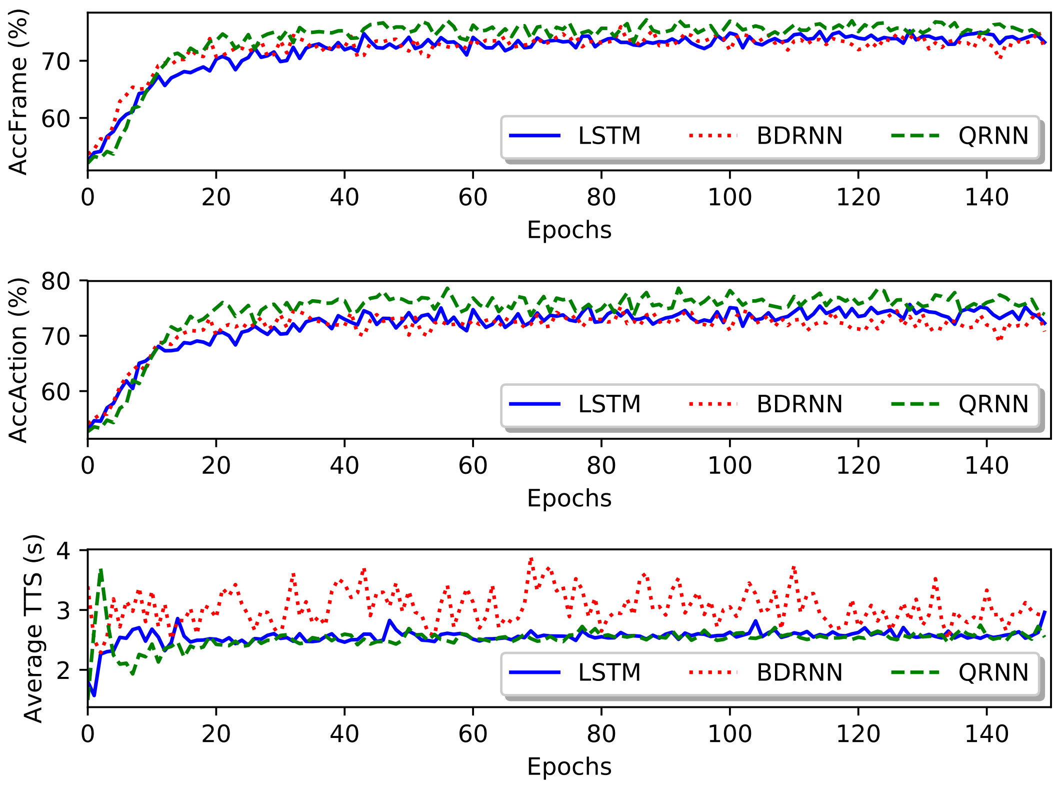


Figure 5.9: Testing performance of green light turn prediction by different models with different training epochs.

Table 5.3: Prediction performance of different models based on their best AccAction.

Task	Model	AccAction (%)	Average TTS (seconds)
LCP	LSTM	89.59 ± 0.41	1.56 ± 0.06
	BDRNN	88.63 ± 0.64	1.25 ± 0.03
	QRNN	**90.52** ± 0.87	**1.50** ± 0.03
GLTP	LSTM	75.65 ± 1.20	2.57 ± 0.11
	BDRNN	75.29 ± 2.06	**2.94** ± 0.57
	QRNN	**78.59** ± 1.72	**2.53** ± 0.06

no obvious change on loss is presented even systems are trained for more epochs

While choosing model weights from training results, two metrics, i.e, AccAction and Average TTS, should be taken into consideration. Table 5.3 shows the best performance achieved by three models. The AccAction for each model is the best among different training epochs, and the Average TTS is the result that corresponds to that epoch. The 2nd-4th rows in Table 5.3 present testing performance of lane change prediction. As can be seen, the QRNN system achieves the best prediction accuracy of 90.52%. This accuracy is 0.93% better than the one achieved by the LSTM system and 1.89% better than the one achieved by the BDRNN system. Meanwhile, the Average TTS of the QRNN system reaches 1.50 seconds and is 0.5 seconds higher than the one achieved by the BDRNN system. This accuracy is also comparable to the best result which is 1.56 seconds, achieved by the LSTM system. Readers may notice the obvious difference between average TTSs in LSTM and BDRNN models implemented by us and the ones achieved in original papers [87] and [88]. This is because the definitions of starting maneuver are different in our work from those in [87] and [88]. We define it as the start time that a car starts yawing, while [87] and [88] define it as the time that a car touches a lane mark. The 5th-7th rows in Table 5.3 present testing performances for turn change prediction at intersections with green lights. Our QRNN system achieves an obvious better accuracy, which is 78.59% (3.3% better than the accuracy achieved by the BDRNN system and 2.9% better than the one by LSTM system); however, the average TTS achieved is not as good as the other two

Table 5.4: Prediction performance of different models based on their best average TTS.

Task	Model	AccAction (%)	Average TTS (seconds)
LCP	LSTM	87.38 ± 2.19	**1.61** ± 0.04
	BDRNN	87.98 ± 0.97	1.34 ± 0.03
	QRNN	**89.17** ± 0.88	**1.54** ± 0.03
GLTP	LSTM	72.26 ± 2.60	2.96 ± 0.53
	BDRNN	72.06 ± 2.50	**3.89** ± 0.65
	QRNN	**75.21** ± 2.02	**2.75** ± 0.20

(0.41 second less than the one in BDRNN).

While Table 5.3 is organized based on the best accuracy of the various models, Table 5.4 is organized based on the best Average TTS of each model. The Average TTS of each model is the best among different training epochs, and the AccAction is the result that corresponds to that epoch. Comparing the 2nd-4th rows in Table 5.4 to the 2nd-4th rows in Table 5.3, we can find an obvious drop in accuracy with a slight increment of Average TTS for each system. The 2nd-4th rows in Table 5.4 show that the proposed QRNN system gives an Average TTS of 1.54 seconds which is only 0.07 seconds less than the best one (1.61 seconds by LSTM). However, our system firmly gives the best accuracy, which is 89.17% (1.19% higher than BDRNN and 1.79% higher LSTM). A similar situation happens to the 5th-7th rows with an exception that the Average TTS achieved by the BDRNN system increases by 0.95 seconds.

The analysis of these two tables illustrates that if model weights are chosen from the epoch that gives the best Average TTS, all models will lose performance. Thus, these weights should be chosen from the epoch with the best prediction accuracy. By this method, the proposed system achieves excellent performance in prediction accuracy and comparable performance in Average TTS.

5.4 Driving Proficiency Assessment by Bidirectional Generalization Ability Test

Driving proficiency assessment is usually performed in three forms: watch on-road test [126], self-questionnaire test [127, 128], or administered cognitive task test [129, 130]. The first and last one require a trained human who has to be on site, thus it is time-consuming. The last two tests are not implemented in real driving scenarios, thus are not considered as direct assessments of drivers' visual, cognitive and decision performance. In-door simulation-based systems are seriously limited by their visual fidelity and only provide specific driving scenarios for assessment [131].

In this work, we propose to estimate driving proficiency through bidirectional generalization ability test of driving maneuver prediction systems (BDGAT). The main intuition behind this method is that feature sequences of maneuvers by drivers with the same driving proficiency should have similar distributions. Therefore, maneuver prediction models trained on drivers with a certain level of driving proficiency should be able to generalize well to other drivers with the same level of proficiency.

As shown in Fig. 5.10, the BDGAT system first acquires a set of canonical drivers' driving records and extracts feature sequences of various maneuvers. With these data sequences, this system trains several maneuver prediction models. These data sequences and trained models are saved for tests on subject drivers. A set of equipment (Cameras, OBD II, etc.) will be installed on a subject driver's personal vehicle. After a period of driving, this driver will be familiar with this equipment and will drive under naturalistic conditions. This set of equipment logs and recognizes a set of maneuvers performed by this driver during daily driving. Feature sequences extracted from these maneuvers are used to train another set of maneuver prediction systems. With two sets of models and two sets of feature sequences, the BDGAT system performs two sets of generalization ability testing. More specifically, the first set, which is named forward testing, is testing models trained with sequences of canonical drivers on feature sequences of the subject driver. The second set (backward testing) is used for testing models trained with sequences of the subject driver on feature sequences of canonical drivers. The subject driver's driving proficiency

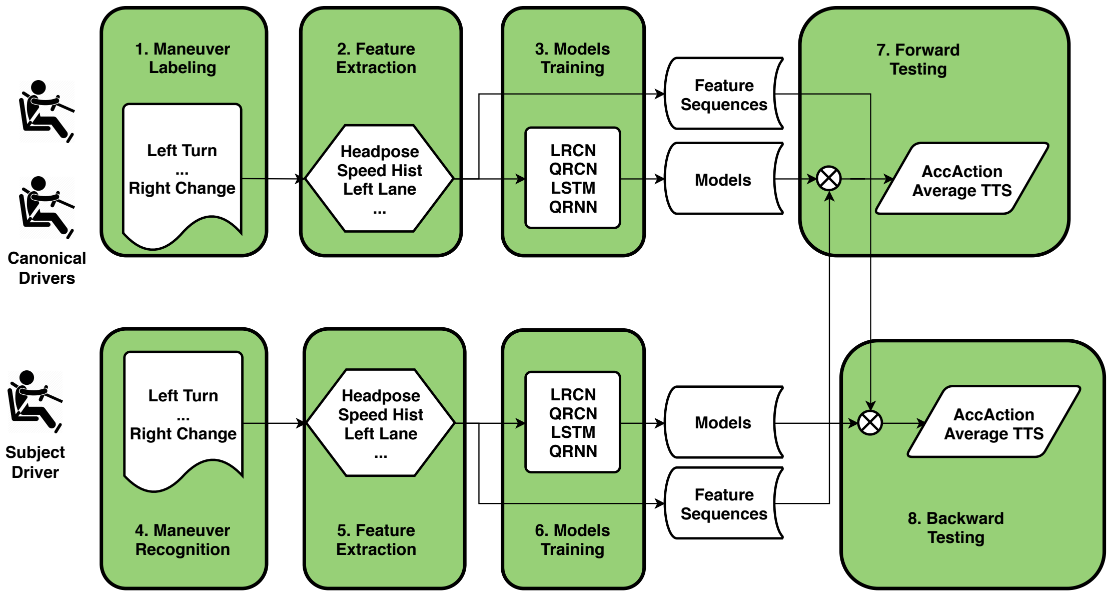


Figure 5.10: An overview of bidirectional generalization ability test system.

is estimated based on the testing results (AccAction) of both directions.

We propose a formulation to calculate a driver's proficiency based on performance, which is

$$\begin{aligned} &Driving\ Proficiency \\ =&\sum_{d}\sum_{t}\sum_{m} r_d * r_t * r_m * (AccAction) \end{aligned} \tag{5.18}$$

where r_d represents the scale of the contribution of the accuracy by testing direction d, r_t represents the scale of the contribution of the accuracy on task t, and r_m represents the scale of the contribution of the accuracy by model m. One significant assumption for machine learning methods is that models trained on a dataset should be able to generalize well on another dataset from the same distribution, and this is bidirectional. Thus, maneuver prediction models trained on driving records of specific drivers (e.g., well trained) should have a generalization ability with respect to similar drivers. The metric to indicate the generalization ability of classification tasks is accuracy. Therefore, the accuracy of models trained and tested on different drivers represents the similarity of these drivers.

Table 5.5: Frequency of blind spot checking for all drivers.

Driver	Blind Spot Checking Frequency (%)			
	Left turn	Right turn	Left change	Right change
1	0.0	0.0	95.45	80.00
2	0.0	0.0	68.42	94.74
3	0.0	0.0	0.0	0.0
4	0.0	0.0	82.35	94.12
5	0.0	0.0	40.00	55.00
6	0.0	0.0	00.00	15.38
7	0.0	0.0	100.00	100.00
Average	**0.0**	**0.0**	**55.17**	**62.75**

Table 5.6: Frequency of mirror checking for all drivers during lane change maneuvers.

Driver	Rear mirror		Side mirror	
	Left change	Right change	Left change	Right change
1	4.55	10.00	100.00	100.00
2	21.05	31.58	100.00	100.00
3	19.05	59.09	100.00	100.00
4	5.88	82.35	100.00	88.24
5	100.00	100.00	96.00	100.00
6	30.77	30.77	100.00	100.00
7	9.09	80.95	100.00	33.33
Average	**27.20**	**56.39**	**99.43**	**88.80**

This accuracy is affected by three factors: the similarity of drivers, the general capacity of models, and the capacity of models on different prediction tasks. The proposed driving proficiency index combines all these factors and a generalization ability should be bidirectional. This index also gives more weight to models with large prediction capacities for different prediction tasks.

To illustrate the effectiveness of the proposed BDGAT, we organize a set of experiments to simulate the testing system based on the driver maneuver data set we collected earlier. The first step is to find canonical drivers. Table 5.5 shows each driver's frequency of blind spots checking among all possible maneuvers. During the whole driving process, Driver 3 never checked the blind spot, while Driver 5 and Driver 6 only checked occasionally. Other drivers check blind spots more often. Table 5.6 presents how frequently drivers check their mirrors during lane change maneuvers. It is obvious that all drivers check side mirrors with high frequency but the frequency of rear mirror checking differs greatly. We rank the average of each driver's frequency of blind spot checking and frequency of rear mirror checking during lane changes ([47.5%, 53.95%, 19.54%, 66.18%, 73.75%, 19.23%, 72.51%] for drivers 1-7.) and choose the four best drivers {2, 4, 5, 7} as canonical drivers to assess drivers {1, 3, 6}. Three models were trained on data for each driver, and the number of training epochs is 100 for all experiments.

Forward testing consists of testing the generalization ability of many canonical drivers on a single driver. This testing result is more robust, thus we set $r_{forward} = 0.6$ and $r_{backward} = 0.4$. As green light turn maneuvers are more complex then lane changes, we set $r_{LCP} = 0.4$ and $r_{GLTP} = 0.6$. For these three models, we set $r_{QRNN} = 0.5$, $r_{LSTM} = 0.3$, and $r_{BDRNN} = 0.2$ as the model that achieves a better prediction performance in Section 5.3.4 gives a more reliable assessment. It is worth noting that these scales are set empirically in this work only because the number of drivers in our data set is limited (we have only 4 canonical drivers). Given a data set of enough drivers, these scales should be set using the following: organize canonical drivers into two sets randomly, use the second set as the testing set and rank drivers' proficiency in this set based on their ground truth proficiency, then adjust these scales so the testing results of BDGAT rank drivers' proficiency correctly.

The testing accuracy for different experiments are shown in Table 5.7. Based on this assessment method, the driving proficiency of driver 1 is 0.78, the driving proficiency of

Table 5.7: Bidirectional testing performance of different models for drivers.

Driver	Task	Model	Forward	Backward
1	GLTP	BDRNN	0.77	0.58
		LSTM	0.80	0.68
		QRNN	0.78	0.58
	LCP	BDRNN	0.95	0.75
		LSTM	0.95	0.76
		QRNN	0.96	0.80
3	GLTP	BDRNN	0.78	0.52
		LSTM	0.74	0.49
		QRNN	0.76	0.51
	LCP	BDRNN	0.97	0.77
		LSTM	0.97	0.82
		QRNN	0.97	0.82
6	GLTP	BDRNN	0.67	0.68
		LSTM	0.76	0.75
		QRNN	0.74	0.75
	LCP	BDRNN	0.73	0.63
		LSTM	0.77	0.67
		QRNN	0.78	0.70

driver 3 is 0.76, and the driving proficiency of driver 6 is 0.73. Meanwhile, according to Table 5.5 and Table 5.6, the average of driver 1's frequency of blind spot checking and frequency of rear mirror checking during lane change maneuvers (47.5%) is higher then that of driver 3 (19.54%), and driver 3 is better than driver 6 (19.23%). These results show that the proposed BDGAT ranks these drivers correctly. However, the difference between driver 1's estimated proficiency and driver 3's estimated proficiency is not larger than that between driver 3 and driver 5, as the size of our dataset constrained us to some extent from fine tuning parameters in proficiency index Equation 5.18.

5.5 Summary

This chapter addresses the question of predicting drivers' imminent maneuvers before they perform actual steering operations. The proposed system uses deep recurrent neural networks to fuse the information regarding driver observation actions and the driving environment. With a new data labeling method and an effective sequential modeling approach, the system is able to predict driving maneuvers with a high accuracy shortly before actual steering operations. A set of experiments show that the proposed approach anticipates lane change maneuvers 1.50 seconds before cars start to yaw with an accuracy improved to 90.52% and anticipates turn maneuvers at intersections with green lights 2.53 seconds before cars start to yaw with an accuracy improved to 78.59%. This chapter also presents a new application for driving maneuver prediction, which is driving proficiency assessment by bidirectional generalization ability test.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The ability to model and understand drivers' behaviors is essential for future ADAS to provide drivers with the appropriate assistance. The proposed research work attempts to investigate new methodologies to design driver behavior recognition and maneuver prediction systems for future ADAS.

We first aim at improving the generalization ability of driver distraction to diverse driving scenarios. To this end, we collect an image dataset of various driving situations from the Internet. Then we introduce Generative Adversarial Networks as a data augmentation method to enhance detection accuracy. The experimental results demonstrate enhanced performance, which we believe originates from more diverse and abundant training samples. In addition, we show the effectiveness of mixing general driving images with images of a specific environment to handle distraction recognition in that specific environment and combining the resultant recognition system as a module in a driver monitoring system. Moreover, this thesis proposes a dual-mode system for recognition under various lighting conditions and designs a new CNN, which can work in real-time on a resources-limited platform, through a novel network bottleneck. The experimental results demonstrate that the proposed system achieves state-of-the-art performance, which arises from the new network architecture with novel bottlenecks for efficient feature representations.

The second part of this thesis focuses on driving maneuver prediction. A new method to label driving maneuver records and a new prediction system are introduced. These novel approaches enhance the prediction performance to the state-of-the-art. In addition, the proposed novel driving proficiency assessment method, as an application of maneuver prediction, demonstrates several advantages against existing methods.

6.2 Future Work

The current work can be extended in many ways, some of which are listed as follows.

- The application scope of the proposed driver distraction recognition system and network is not limit to distraction activities related to electronic devices only. A promising future direction is to extend the current work to include comprehensive images from real driving environments and to test the system for daily driving scenarios.
- Furthermore, the proposed network architecture may be incorporated into other computer vision applications in computational resource-limited environments. For example, there is an increasing trend to enhance the recognition ability of small-sized robots.
- The design idea of the proposed driving monitoring system can be extended to many other applications. Take a system to warn workers trying to touch a moving mechanical part in a factory as an example. The required basic steps will be camera position design, data collection, network design, and system deployment. The camera should be installed in a position where is able to collect clear images of workers' hands. With a camera installed, the next step is to collect an image data of different workers trying to touch that part. Images from other resources, for example the Internet, with similar human positions or similar scenarios could be used for training. The next step is network training and optimization, so that it is small enough to be deployed on small computational devices with limited resources.
- Our work on driving maneuver prediction can be extended to other maneuvers that involve various traffic conditions. For example, a large number of intersections are

controlled by stop signs in North America. Collecting a comprehensive data set of these intersections and exploring effective prediction methods would be an interesting direction yet to be explored.

- Another direction is building a comprehensive and naturalistic data set for driver maneuver prediction, as the sizes of current data sets are still small compared to the complexity of the addressed problem. Meanwhile, the absence of benchmark data set limits the comparisons on various methods proposed by different research groups.

References

[1] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016.

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

[3] Pytorch. Torchvision Models. https://pytorch.org/docs/stable/torchvision/models.html/, 2018. [Online; accessed 18-April-2019].

[4] Eric Sun. Shufflenet-v2-Pytorch. https://github.com/ericsun99/Shufflenet-v2-Pytorch/, 2018. [Online; accessed 18-March-2019].

[5] Evgeniy Zheltonozhskiy. Impementation of MobileNetV2 in pytorch. https://github.com/Randl/MobileNetV2-pytorch/, 2018. [Online; accessed 18-March-2019].

[6] UDRIVE. European drivers spend 10% of the ride on secondary tasks. http://www.udrive.eu/index.php/news/145-european-drivers-spend-10-of-the-ride-on-secondary-tasks. Accessed: 2018-07-18.

[7] Thomas A Dingus, Feng Guo, Suzie Lee, Jonathan F Antin, Miguel Perez, Mindy Buchanan-King, and Jonathan Hankey. Driver crash risk factors and prevalence evaluation using naturalistic driving data. *Proceedings of the National Academy of Sciences*, 113(10):2636–2641, 2016.

[8] National Research Council. Strategic highway research: Saving lives, reducing congestion, improving quality of life. Technical report, National Research Council, 2001.

[9] Yvonne Barnard, Fabian Utesch, Nicole Nes, Rob Eenink, and Martin Baumann. The study design of UDRIVE: The naturalistic driving study across europe for cars, trucks and scooters. *Springer European Transport Research Review*, 8(2):1–10, 2016.

[10] MA Regan, Ann Williamson, Raphael Grzebieta, Judith Charlton, M Lenneb, Barry Watson, Narelle Haworth, Andry Rakotonirainy, Jeremy Woolley, and Robert Anderson. The Australian 400-car naturalistic driving study: Innovation in road safety research and policy. In *Australasian Road Safety Research, Policing & Education Conference*, pages 1–13, 2013.

[11] Andrea Heide and Klaus Henning. The "cognitive car": A roadmap for research issues in the automotive sector. *Elsevier Annual Reviews in Control*, 30(2):197–203, 2006.

[12] Li Li, Ding Wen, Nan-Ning Zheng, and Lin-Cheng Shen. Cognitive cars: A new frontier for ADAS research. *IEEE Transactions on Intelligent Transportation Systems*, 13(1):395–407, 2012.

[13] Yujun Zeng, Xin Xu, Dayong Shen, Yuqiang Fang, and Zhipeng Xiao. Traffic sign recognition using kernel extreme learning machines with deep perceptual features. *IEEE Transactions on Intelligent Transportation Systems*, 18(6):1647–1653, 2017.

[14] Xiao Lu, Yaonan Wang, Xuanyu Zhou, Zhenjun Zhang, and Zhigang Ling. Traffic sign recognition via multi-modal tree-structure embedded multi-task learning. *IEEE Transactions on Intelligent Transportation Systems*, 18(4):960–972, 2017.

[15] SAE On-Road Automated Vehicle Standards Committee. Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems. Technical report, SAE, 2014.

[16] Tomonori Mizushima, Pongsathorn Raksincharoensak, and Masao Nagai. Direct yaw-moment control adapted to driver behavior recognition. In *IEEE International Joint Conference on SICE-ICASE*, pages 534–539, 2006.

[17] Yuichi Saito, Makoto Itoh, and Toshiyuki Inagaki. Driver assistance system with a dual control scheme: Effectiveness of identifying driver drowsiness and preventing lane departure accidents. *IEEE Transactions on Human-Machine Systems*, 46(5):660–671, 2016.

[18] Duy Tran, Eyosiyas Tadesse, Weihua Sheng, Yuge Sun, Meiqin Liu, and Senlin Zhang. A driver assistance framework based on driver drowsiness detection. In *IEEE International Conference on CYBER Technology in Automation, Control, and Intelligent Systems*, pages 173–178, 2016.

[19] Ashesh Jain, Avi Singh, Hema S Koppula, Shane Soh, and Ashutosh Saxena. Recurrent neural networks for driver activity anticipation via sensory-fusion architecture. In *IEEE International Conference on Robotics and Automation*, pages 3118–3125, 2016.

[20] Seong-Woo Kim and Wei Liu. Cooperative autonomous driving: A mirror neuron inspired intention awareness and cooperative perception approach. *IEEE Intelligent Transportation Systems Magazine*, 8(3):23–32, 2016.

[21] Pamela I. Labuhn and William J. Chundrlik Jr. Adaptive cruise control, October 3 1995. US Patent 5,454,442.

[22] Deaglan O'meachair and Matthew Crumpton. Adaptive braking system and method, April 4 2017. US Patent 9,610,931.

[23] Hajime Oyama. Lane keeping control system for vehicle, January 5 2016. US Patent 9,227,634.

[24] Yoshio Mukaiyama. Automatic parking system, April 12 2016. US Patent 9,308,913.

[25] Marco J. Flores, José M. Armingol, and Arturo de la Escalera. Driver drowsiness warning system using visual information for both diurnal and nocturnal illumination conditions. *EURASIP Journal on Advances in Signal Processing*, 2010(1):438205, 2010.

[26] Nuria Oliver and Alex P. Pentland. Graphical models for driver behavior recognition in a smartcar. In *IEEE Intelligent Vehicles Symposium*, pages 7–12, 2000.

[27] Thomas B. Sheridan. Driver distraction from a control theory perspective. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 46(4):587–599, 2004.

[28] Myounghoon Jeon and Bruce N. Walker. What to detect? Analyzing factor structures of affect in driving contexts for an emotion detection and regulation system. In *Sage the Human Factors and Ergonomics Society Annual Meeting*, volume 55, pages 1889–1893, 2011.

[29] Michael A Regan, John D Lee, and Kristie Young. *Driver distraction: Theory, effects, and mitigation.* CRC Press, 2008.

[30] Sheila G Klauer, Feng Guo, Bruce G Simons-Morton, Marie Claude Ouimet, Suzanne E Lee, and Thomas A Dingus. Distracted driving and risk of road crashes among novice and experienced drivers. *New England Journal of Medicine*, 370(1):54–59, 2014.

[31] Yusuf Artan, Orhan Bulan, Robert Loce, and Peter Paul. Driver cell phone usage detection from HOV/HOT NIR images. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 225–230, 2014.

[32] Keshav Seshadri, Felix Juefei-Xu, Dipan Pal, Marios Savvides, and Craig Thor. Driver cell phone usage detection on strategic highway research program (SHRP2) face view videos. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 35–43, 2015.

[33] Jie Yang, Simon Sidhom, Gayathri Chandrasekaran, Tam Vu, Hongbo Liu, Nicolae Cecan, Yingying Chen, Marco Gruteser, and Richard P. Martin. Detecting driver phone use leveraging car speakers. In *ACM International Conference on Mobile Computing and Networking*, pages 97–108, 2011.

[34] Tiffany L Overton, Terry E Rives, Carrie Hecht, Shahid Shafi, and Rajesh R Gandhi. Distracted driving: prevalence, problems, and prevention. *International Journal of Injury Control and Safety Promotion*, 22(3):187–192, 2015.

[35] Matti Kutila, Maria Jokela, Gustav Markkula, and Maria R. Rué. Driver distraction detection with a camera vision system. In *IEEE International Conference on Image Processing*, volume 6, pages 201–204, 2007.

[36] Yulan Liang, Michelle L. Reyes, and John D. Lee. Real-time detection of driver cognitive distraction using support vector machines. *IEEE Transactions on Intelligent Transportation Systems*, 8(2):340–350, 2007.

[37] Masahiro Miyaji, Mikio Danno, Haruki Kawanaka, and Koji Oguri. Drivers cognitive distraction detection using adaboost on pattern recognition basis. In *IEEE International Conference on Vehicular Electronics and Safety*, pages 51–56, 2008.

[38] T. Hoang Ngan Le, Yutong Zheng, Chenchen Zhu, Khoa Luu, and Marios Savvides. Multiple scale faster-RCNN approach to driver's cell-phone usage and hands on steering wheel detection. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 46–53, 2016.

[39] Tashrif Billah, SM Mahbubur Rahman, M Omair Ahmad, and MNS Swamy. Recognizing distractions for assistive driving by tracking body parts. *IEEE Transactions on Circuits and Systems for Video Technology*, 2018.

[40] Martin Wollmer, Christoph Blaschke, Thomas Schindl, Björn Schuller, Berthold Farber, Stefan Mayer, and Benjamin Trefflich. Online driver distraction detection using Long Short-Term Memory. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):574–582, 2011.

[41] Nanxiang Li, Jinesh J. Jain, and Carlos Busso. Modeling of driver behavior in real world scenarios using multiple noninvasive sensors. *IEEE Transactions on Multimedia*, 15(5):1213–1225, 2013.

[42] Nanxiang Li and Carlos Busso. Predicting perceived visual and cognitive distractions of drivers with multimodal features. *IEEE Transactions on Intelligent Transportation Systems*, 16(1):51–65, 2015.

[43] Nanxiang Li and Carlos Busso. Analysis of facial features of drivers under cognitive and visual distractions. In *IEEE International Conference on Multimedia and Expo*, pages 1–6, 2013.

[44] E Friesen and Paul Ekman. Facial action coding system: a technique for the measurement of facial movement. *Palo Alto*, 3, 1978.

[45] Masahiro Miyaji, Haruki Kawanaka, and Koji Oguri. Driver's cognitive distraction detection using physiological features by the adaboost. In *IEEE International Conference on Intelligent Transportation Systems*, pages 1–6, 2009.

[46] Cheng Bo, Xuesi Jian, Xiang-Yang Li, Xufei Mao, Yu Wang, and Fan Li. You're driving and texting: detecting drivers using personal smart phones by leveraging inertial sensors. In *Annual International Conference on Mobile Computing and Networking*, pages 199–202. ACM, 2013.

[47] Zhaojian Li, Shan Bao, Ilya V Kolmanovsky, and Xiang Yin. Visual-manual distraction detection using driving performance indicators with naturalistic driving data. *IEEE Transactions on Intelligent Transportation Systems*, 19(8):2528–2535, 2017.

[48] Seyed Mehdi Iranmanesh, Hossein Nourkhiz Mahjoub, Hadi Kazemi, and Yaser P Fallah. An adaptive forward collision warning framework design based on driver distraction. *IEEE Transactions on Intelligent Transportation Systems*, 19(12):3925–3934, 2018.

[49] Andrei Aksjonov, Pavel Nedoma, Valery Vodovozov, Eduard Petlenkov, and Martin Herrmann. Detection and evaluation of driver distraction using machine learning and fuzzy logic. *IEEE Transactions on Intelligent Transportation Systems*, 20(6):2048–2059, 2019.

[50] Yuan Liao, Shengbo Eben Li, Wenjun Wang, Ying Wang, Guofa Li, and Bo Cheng. Detection of driver cognitive distraction: A comparison study of stop-controlled intersection and speed-limited highway. *IEEE Transactions on Intelligent Transportation Systems*, 17(6):1628–1637, 2016.

[51] Tianchi Liu, Yan Yang, Guang-Bin Huang, Yong Kiang Yeo, and Zhiping Lin. Driver distraction detection using semi-supervised machine learning. *IEEE Transactions on Intelligent Transportation Systems*, 17(4):1108–1120, 2015.

[52] Yang Xing, Chen Lv, Huaji Wang, Dongpu Cao, Efstathios Velenis, and Fei-Yue Wang. Driver activity recognition for intelligent vehicles: A deep learning approach. *IEEE Transactions on Vehicular Technology*, 2019.

[53] Xuetao Zhang, Nanning Zheng, Fei Wang, and Yongjian He. Visual recognition of driver hand-held cell phone use based on hidden CRF. In *IEEE International Conference on Vehicular Electronics and Safety*, pages 248–251, 2011.

[54] Amira Ragab, Celine Craye, Mohamed S. Kamel, and Fakhri Karray. A visual-based driver distraction recognition and detection using random forest. In *Springer International Conference Image Analysis and Recognition*, pages 256–265, 2014.

[55] Arief Koesdwiady, Safaa M Bedawi, Chaojie Ou, and Fakhri Karray. End-to-end deep learning for driver distraction recognition. In *International Conference Image Analysis and Recognition*, pages 11–18. Springer, 2017.

[56] Hesham M Eraqi, Yehya Abouelnaga, Mohamed H Saad, and Mohamed N Moustafa. Driver distraction identification with an ensemble of convolutional neural networks. *Journal of Advanced Transportation*, 2019, 2019.

[57] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[58] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[59] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[60] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[61] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5MB model size. *arXiv preprint arXiv:1602.07360*, 2016.

[62] Jonghoon Jin, Aysegul Dundar, and Eugenio Culurciello. Flattened convolutional neural networks for feedforward acceleration. *arXiv preprint arXiv:1412.5474*, 2014.

[63] Min Wang, Baoyuan Liu, and Hassan Foroosh. Factorized convolutional neural networks. In *IEEE International Conference on Computer Vision*, pages 545–553, 2017.

[64] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[65] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobilenetV2: Inverted residuals and linear bottlenecks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[66] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018.

[67] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet V2: Practical guidelines for efficient CNN architecture design. In *European Conference on Computer Vision*, pages 116–131, 2018.

[68] Laurent Sifre and Stéphane Mallat. Rigid-motion scattering for image classification. *PhD thesis, Ph. D. thesis*, 1:3, 2014.

[69] Arief Koesdwiady, Ridha Soua, Fakhreddine Karray, and Mohamed S Kamel. Recent trends in driver safety monitoring systems: State of the art and challenges. *IEEE Transactions on Vehicular Technology*, 66(6):4550–4563, 2016.

[70] Yang Xing, Chen Lv, Huaji Wang, Hong Wang, Yunfeng Ai, Dongpu Cao, Efstathios Velenis, and Fei-Yue Wang. Driver lane change intention inference for intelligent vehicles: Framework, survey, and challenges. *IEEE Transactions on Vehicular Technology*, 68(5):4377–4390, 2019.

[71] Jürgen Wiest, Matthias Karg, Felix Kunz, Stephan Reuter, Ulrich Kreßel, and Klaus Dietmayer. A probabilistic maneuver prediction framework for self-learning vehicles with application to intersections. In *IEEE Intelligent Vehicles Symposium*, pages 349–355, 2015.

[72] Peng Liu and Arda Kurt. Trajectory prediction of a lane changing vehicle based on driver behavior estimation and classification. In *IEEE International Conference on Intelligent Transportation Systems*, pages 942–947, 2014.

[73] Anup Doshi and Mohan M. Trivedi. On the roles of eye gaze and head dynamics in predicting driver's intent to change lanes. *IEEE Transactions on Intelligent Transportation Systems*, 10(3):453–462, 2009.

[74] Stéphanie Lefèvre, Christian Laugier, and Javier Ibañez-Guzmán. Exploiting map information for driver intention estimation at road intersections. In *IEEE Intelligent Vehicles Symposium*, pages 583–588, 2011.

[75] Michaël G. Ortiz, Jens Schmüdderich, Franz Kummert, and Alexander Gepperth. Situation-specific learning for ego-vehicle behavior prediction systems. In *IEEE International Conference on Intelligent Transportation Systems*, pages 1237–1242, 2011.

[76] Tran Cuong, Anup Doshi, and Mohan Manubhai Trivedi. Modeling and prediction of driver behavior by foot gesture analysis. *Elsevier Computer Vision and Image Understanding*, 116(3):435–445, 2012.

[77] Cuong Tran, Anup Doshi, and Mohan M. Trivedi. Pedal error prediction by driver foot gesture analysis: A vision-based inquiry. In *IEEE Intelligent Vehicles Symposium*, pages 577–582, 2011.

[78] Zachary N Sunberg, Christopher J Ho, and Mykel J Kochenderfer. The value of inferring the internal state of traffic participants for autonomous freeway driving. In *IEEE American Control Conference*, pages 3004–3010, 2017.

[79] Vijay Gadepally, Ashok Krishnamurthy, and Umit Ozguner. A framework for estimating driver decisions near intersections. *IEEE Transactions on Intelligent Transportation Systems*, 15(2):637–646, 2014.

[80] Georges S. Aoude, Vishnu R. Desaraju, Lauren H. Stephens, and Jonathan P. How. Driver behavior classification at intersections and validation on large naturalistic data set. *IEEE Transactions on Intelligent Transportation Systems*, 13(2):724–736, 2012.

[81] Ryuki Mabuchi and Keiichi Yamada. Prediction of driver's stop or go at yellow traffic signal from vehicle behavior. In *IEEE Intelligent Vehicles Symposium*, pages 1161–1166, 2013.

[82] Kim Schmidt, Matthias Beggiato, Karl H. Hoffmann, and Josef F. Krems. A mathematical model for predicting lane changes using the steering wheel angle. *Elsevier Journal of Safety Research*, 49(2014):85–90, 2014.

[83] Brendan Morris, Anup Doshi, and Mohan Trivedi. Lane change intent prediction for driver assistance: On-road design and evaluation. In *IEEE Intelligent Vehicles Symposium*, pages 895–901, 2011.

[84] Michaël Garcia Ortiz, Franz Kummert, and Jens Schmüdderich. Prediction of driver behavior on a limited sensory setting. In *IEEE International Conference on Intelligent Transportation Systems*, pages 638–643, 2012.

[85] Puneet Kumar, Mathias Perrollaz, Stéphanie Lefevre, and Christian Laugier. Learning-based approach for online lane change intention prediction. In *IEEE Intelligent Vehicles Symposium*, pages 797–802, 2013.

[86] Julian Schlechtriemen, Andreas Wedel, Joerg Hillenbrand, Gabi Breuel, and Klaus-Dieter Kuhnert. A lane change detection approach using feature ranking with maximized predictive power. In *IEEE Intelligent Vehicles Symposium*, pages 108–114, 2014.

[87] Ashesh Jain, Hema S Koppula, Shane Soh, Bharad Raghavan, Avi Singh, and Ashutosh Saxena. Brain4cars: Car that knows before you do via sensory-fusion deep learning architecture. *arXiv preprint arXiv:1601.00740*, 2016.

[88] Oluwatobi Olabiyi, Eric Martinson, Vijay Chintalapudi, and Rui Guo. Driver action prediction using deep (bidirectional) Recurrent Neural Network. *arXiv preprint arXiv:1706.02257*, 2017.

[89] Ashesh Jain, Hema S. Koppula, Bharad Raghavan, Shane Soh, and Ashutosh Saxena. Car that knows before you do: Anticipating maneuvers via learning temporal driving models. In *IEEE International Conference on Computer Vision*, pages 3182–3190, 2015.

[90] Julia Nilsson, Jonas Fredriksson, and Erik Coelingh. Rule-based highway maneuver intention recognition. In *IEEE International Conference on Intelligent Transportation Systems*, pages 950–955, 2015.

[91] Arief Koesdwiady, Ramzi Abdelmoula, Fakhri Karray, and Mohamed Kamel. Driver inattention detection system: A pso-based multiview classification approach. In *IEEE International Conference on Intelligent Transportation Systems*, pages 1624–1629, 2015.

[92] Georgios K Kountouriotis, Panagiotis Spyridakos, Oliver MJ Carsten, and Natasha Merat. Identifying cognitive distraction using steering wheel reversal rates. *Accident Analysis & Prevention*, 96:39–45, 2016.

[93] Tianchi Liu, Yan Yang, Guang-Bin Huang, Yong Kiang Yeo, and Zhiping Lin. Driver distraction detection using semi-supervised machine learning. *IEEE Transactions on Intelligent Transportation Systems*, 17(4):1108–1120, 2016.

[94] Kede Ma, Wentao Liu, Kai Zhang, Zhengfang Duanmu, Zhou Wang, and Wangmeng Zuo. End-to-end blind image quality assessment using deep neural networks. *IEEE Transactions on Image Processing*, 27(3):1202–1213, 2018.

[95] Kede Ma, Huan Fu, Tongliang Liu, Zhou Wang, and Dacheng Tao. Local blur mapping: Exploiting high-level semantics by deep neural networks. *arXiv preprint arXiv:1612.01227*, 2016.

[96] Yehya Abouelnaga, Hesham M Eraqi, and Mohamed N Moustafa. Real-time distracted driver posture classification. *arXiv preprint arXiv:1706.09498*, 2017.

[97] Kaggle. State farm distracted driver detection. https://www.kaggle.com/c/state-farm-distracted-driver-detection. Accessed: 2018-06-11.

[98] Salah Taamneh, Panagiotis Tsiamyrtzis, Malcolm Dcosta, Pradeep Buddharaju, Ashik Khatri, Michael Manser, Thomas Ferris, Robert Wunderlich, and Ioannis Pavlidis. A multimodal dataset for various forms of distracted driving. *Scientific Data*, 4:170110, 2017.

[99] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[100] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[101] Chaojie Ou, Chahid Ouali, Safaa M Bedawi, and Fakhri Karray. Driver behavior monitoring using tools of deep learning and fuzzy inferencing. In *2018 IEEE International Conference on Fuzzy Systems*, pages 1–7, 2018.

[102] Anders Krogh and John A. Hertz. A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems*, pages 950–957, 1992.

[103] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[104] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[105] Martin Arjovsky. Code accompanying the paper "wasserstein gan". https://github.com/martinarjovsky/WassersteinGAN. Accessed: 2017-12-30.

[106] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[107] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[108] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[109] Tadas Baltrušaitis, Peter Robinson, and Louis-Philippe Morency. Openface: an open source facial behavior analysis toolkit. In *IEEE Conference on Applications of Computer Vision*, pages 1–10, 2016.

[110] Tadas Baltrusaitis, Peter Robinson, and Louis-Philippe Morency. Constrained local neural fields for robust facial landmark detection in the wild. In *IEEE International Conference on Computer Vision Workshops*, pages 354–361, 2013.

[111] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[112] Mercury Insurance. Distracted Driving Isnt Worth the Risk. https://blog.mercuryinsurance.com/insurance/distracted-driving-isnt-worth-the-risk/, 2018. [Online; accessed 18-April-2019].

[113] GovAppSolutions. Distracted Driving: Day and Night. https://govappsolutions.com/distracted-driving-day-and-night//, 2018. [Online; accessed 18-April-2019].

[114] Murtadha D Hssayeni, Sagar Saxena, Raymond Ptucha, and Andreas Savakis. Distracted driver detection: Deep learning vs handcrafted features. *Electronic Imaging*, 2017(10):20–26, 2017.

[115] Government of Ontario. What counts as distracted driving. https://www.ontario.ca/page/distracted-driving. Accessed: 2019-03-12.

[116] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1251–1258, 2017.

[117] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[118] National Highway Traffic Safety Administration. National motor vehicle crash causation survey: Report to congress. Technical report, National Highway Traffic Safety Administration, 2008.

[119] Santokh Singh. Critical reasons for crashes investigated in the national motor vehicle crash causation survey. Technical report, National Highway Traffic Safety Administration, 2015.

[120] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[121] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, 2013.

[122] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. Hybrid speech recognition with deep bidirectional LSTM. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 273–278, 2013.

[123] James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. Quasi-Recurrent Neural Networks. In *International Conference on Learning Representations*, 2017.

[124] Muhammad Abdul-Mageed and Lyle Ungar. Emonet: Fine-grained emotion detection with gated recurrent neural networks. In *Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 718–728, 2017.

[125] Chen Li, Zhen Zhang, Wee Sun Lee, and Gim Hee Lee. Convolutional sequence to sequence model for human dynamics. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5226–5234, 2018.

[126] Driveable Assessment Centres. The driveable on-road evaluation. `https://driveable.com/index.php/products/dore`. Accessed: 2019-2-15.

[127] AAA SENIOR DRIVING. Drivers 65 plus: Check your performance. `https://seniordriving.aaa.com/evaluate-your-driving-ability/self-rating-tool/`. Accessed: 2018-12-10.

[128] David W Eby, Lisa J Molnar, Jean T Shope, Jonathon M Vivoda, and Tiffani A Fordyce. Improving older driver knowledge and self-awareness through self-assessment: The driving decisions workbook. *Journal of Safety Research*, 34(4):371–381, 2003.

[129] CogniFit. Online driving test: Cognitive assessment battery (dab). `https://www.cognifit.com/cognitive-assessment/driving-test`. Accessed: 2018-12-10.

[130] Driveable Assessment Centres. The driveable cognitive assessment tool. https://driveable.com/index.php/products/dcat. Accessed: 2019-2-15.

[131] Tomoaki Nakano, Muneo Yamada, and Shin Yamamoto. A method for assessing the driving ability of the elderly and thoughts on its systematization. *IATSS research*, 32(1):44–53, 2008.

APPENDICES

1. Two videos shown below present the working process of the driver monitoring system proposed in Section4, Chapter 3.
2. This video (https://www.dropbox.com/s/rryfle8q0y5afof/speed20.avi?dl=0) shows the system working while a driver is driving on a speed of 20 km/hour. In this video, the estimated danger level stays at 0 while he/she is driving normally (two hands on the steering wheel while watching hi front). The danger level starts to increase once he/she starts using a phone(texting or talking) or keeps watching his/her right side. Instantly after he returns back to normal driving, the danger level drops to zero.
3. This video (https://www.dropbox.com/s/z0fl4h056hg6ws5/speed100.avi?dl=0) shows the system working while a driver is driving on a speed of 100 km/hour. In this video, the estimated danger level increases or decreases more sharply than it is with a speed of 20.