University of Hertfordshire
School of Engineering and Technology

Detection of Driver Drowsiness and Distraction using Computer Vision and Machine Learning Approaches

Ofonime Dominic Okon

Submitted to the University of Hertfordshire in partial fulfilment
of the requirements for the degree of Master of Philosophy (MPhil)

November 2018

Abstract

Drowsiness and distracted driving are leading factor in most car crashes and near-crashes. This research study explores and investigates the applications of both conventional computer vision and deep learning approaches for the detection of drowsiness and distraction in drivers.

In the first part of this MPhil research study conventional computer vision approaches was studied to develop a robust drowsiness and distraction system based on yawning detection, head pose detection and eye blinking detection. These algorithms were implemented by using existing human crafted features. Experiments were performed for the detection and classification with small image datasets to evaluate and measure the performance of system. It was observed that the use of human crafted features together with a robust classifier such as SVM gives better performance in comparison to previous approaches. Though, the results were satisfactorily, there are many drawbacks and challenges associated with conventional computer vision approaches, such as definition and extraction of human crafted features, thus making these conventional algorithms to be subjective in nature and less adaptive in practice.

In contrast, deep learning approaches automates the feature selection process and can be trained to learn the most discriminative features without any input from human. In the second half of this research study, the use of deep learning approaches for the detection of distracted driving was investigated. It was observed that one of the advantages of the applied methodology and technique for distraction detection includes and illustrates the contribution of CNN enhancement to a better pattern recognition accuracy and its ability to learn features from various regions of a human body simultaneously. The comparison of the performance of four convolutional deep net architectures (AlexNet, ResNet, MobileNet and NASNet) was carried out, investigated triplet training and explored the impact of combining a support vector classifier (SVC) with a trained deep net. The images used in our experiments with the deep nets are from the State Farm Distracted Driver Detection dataset hosted on Kaggle, each of which captures the entire body of a driver. The best results were obtained with the NASNet trained using triplet loss and combined with an SVC. It was observed that one of the advantages of deep learning approaches are their ability to learn discriminative features from various regions of a human body simultaneously. The ability has enabled deep learning approaches to reach accuracy at human level.

Acknowledgement

I am immensely thankful to God Almighty, in which in his mercy and loving grace, has protected me and blessed me with wisdom and knowledge to complete this project and research study. He shielded me with love, mercy, and from woes, during my stay at the University. I am eternally grateful and appreciative to him.

With gratitude, I would like to thank my supervisor, Dr. Lily Meng, for her wisdom, ongoing support, attentiveness and impartial advice, whenever I was faced with issues, concerns or challenges during the development and final stages of this project. Thank you for guiding and reinforcing me with patience and unflinching support while on my way to success.

While creating and developing my project, I was blessed and privileged to have continuous support from my dear father, Obong Dominic Edet Okon Umoetok. Thank you for providing me with the much-needed support, advice and also for helping me to see things in perspective. May the good Lord continue to bless you. To my mother, Late Mrs. Dominica Dominic Okon, which bestowed upon on me the motivation and patience to carry out my project successfully. She was a blessing to me and may her soul rest in peace.

Furthermore, I would like to thank my siblings, Irene, Dorathy, Anthonia, Pius and Dominica Jnr. for their generous assistance and encouragement. May God bless all of you. I would also like to take this opportunity to thank with recognition, to all members of my family and friends, who has granted me with the inspiration, patience and tranquility during the competition of my project. May God bless you all.

Dedication

I dedicate this project to God Almighty, my father Obong Dominic Edet Okon Umoetok, and also my dear mother, late Mrs. Dominica Dominic Okon.

Declaration

DECLARATION STATEMENT

I, Ofonime Dominic Okon, the author of this project, hereby declare that this research study titled "Detection of Driver Drowsiness and Distraction using Computer Vision and Machine Learning Approaches" is my own genuine work from beginning to end. All the materials, sources, information and research used in this thesis, were correctly and satisfactorily acknowledged by means of IEEE Numeric Referencing and Harvard System of Referencing. (ref. UPRAS/C/6.1, Appendix I, Section 2 – Section on cheating and plagiarism)

Student Full Name: Ofonime Dominic Okon

Student Registration Number: 14179916

Signed:

…….

Date: 05/11/2018

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AAAFTS	American Automobile Association Foundation for Traffic Safety
Adam	Adaptive Moment Estimation
ADAS	Advanced Driver Assistance System
AECS	Average Eye Closure Speed
BAIR	Berkley AI Research
BHE	Bi-Histogram Equalization
BU-3DFE	Birmingham University 3D Facial Expression
CCD	Charge-Coupled Device
CNN	Convolutional Neural Network
DBN	Deep Belief Networks
DDD	Deep Drowsiness Detection
DOO	Degree of Openness
FC	Fully Connected
FF-Bp	Feed Forward Back Propagation
GPU	Graphical Processing Unit
HMM	Hidden Markov Model
HOG	Histograms Oriented Gradient
LBP	Local Binary Patterns
LDA	Linear Discriminant Analysis
LRN	Local Response Normalization
MAC	Multiply and Accumulated
MLP	Multiple Layer Perceptron
MSE	Mean Squared Error
NAS	Neural Architecture Search
NHTSA	National Highway Traffic Safety Administration
NIR	Near Infrared

NLL	Negative Log Likelihood
PCA	Principal Component Analysis
PERCLOS	Percentage Eye Closure
PRC	Percentage Road Centre
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RI	Region of Interest
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SIFT	Scale Invariant Feature Transform
SVC	Support Vector Classifier
SVM	Support Vector Machine
SVR	Support Vector Regressor

CHAPTER 1

INTRODUCTION

1.1. Introduction and Background

The integration of advanced safety systems in modern vehicles has reduced the number of road accidents significantly. However, the number of accidents are still not under an acceptable range. In their research, Murray and Lopez [1] reported that road accidents would be the third main reason for deaths by the year 2020. Fatigue and drowsiness among drivers result in fatal driving errors are considered one of the most important causes of road accidents [2, 3]. The involvement of the number of factors such as driver attention, cognitive skills and physical fitness makes driving a relatively complex task [4]. However, it is not uncommon for drivers to involve themselves in activities that distract their attention from driving tasks. These distraction-related activities significantly degrade the driving performance and, due to the complex nature of driving, result in road accidents. Some common distraction-related activities of drivers include conversations with other passengers, the use of technological devices (mobile phones, navigation systems, radios, etc.), eating and using makeup tools while driving [5]. According to a report published by National Highway Traffic Safety Administration (NHTSA), of all the road accidents, approximately 25% are due to the inattention of drivers [6], and half of those 25% are due to the distraction of drivers [7, 8].

Fatigue and drowsiness are the hypnosis effects among drivers, and the major causes of these effects include lack of sleep, long and continuous drives, illness and the use of drugs [9]. Although the introduction of advanced comfort level and autonomy in modern day vehicles has improved the safety of drivers, it also contributes to fatigue and drowsiness of drivers [10, 11]. In a study, Sagberg et al. [11] reported that drowsy driving increased the probability of errors in driving due to drivers' impaired mental capacity. Fatigue and drowsiness-related accidents are considered more fatal and more dangerous compared to normal accidents because of their direct effects on the decision-making abilities of drivers. Fatigue and drowsiness are technically different. Fatigue is defined as the extreme tiredness

because of physical and mental exertions when a person is executing some tasks, whereas drowsiness is the state resulted by lack of rest and sleep [3]. However, fatigue and drowsiness are often interchangeable. In many existing driver monitoring systems, both terms are used for the same meaning as the symptoms of both health conditions are almost identical. Hence, in this research, the term of drowsiness will be used to represent the mutual effect of fatigue and drowsiness on driving performance.

In the research carried out by Lal and Craig [12], they pointed out a number of symptoms that could facilitate the detection of drivers fatigue, including eye blinking rate, yawning frequency, variations in mouth positions, variations in head positions and driving patterns. In a report by Federal Motor Carrier Safety Administration [13], the relationship between the number of continuous driving hours and fatigue-related accidents has been indicated and stated that an increase in continuous driving hours increased the percentage of fatigue-related accidents (up to 4 percent increase for 10 continuous hours of driving). Kaggle challenge defined nine types of distracted driving.

According to Bayly et al. [14], the number of road accidents can be reduced by about 20% if there is a proper mechanism that can monitor in-drive behaviours of drivers. The development of efficient driver-attention monitoring systems using the state-of-the-art emerging technologies is one possible solution that can reduce the number of accidents and improve road safety. Dinges and Mallis [15] listed four types of drowsiness detection approaches: the mathematical model-based approach, fitness for duty technologies approach, vehicle performance-based approach and in-vehicle operator monitoring-based approach. The in-vehicle operator monitoring based approach is widely explored by researchers, which uses computer vision-based technologies and studies the physiological signals to detect the attention level of drivers. The computer vision-based approach is non-intrusive and easily applicable in real-time situations. Furthermore, the detection results of these approaches are more effective and adaptive than others; hence, they are active areas for researchers in this domain. In addition to drowsiness, distraction among drivers is equally significant while developing attention-monitoring systems. In one study, Thimbleby et al. [16] indicated how in-vehicle objects could distract drivers and affect their driving performance.

State Farm has taken an initiative to improve road safety by introducing an initiative called Kaggle challenge for detection of distracted drivers, where camera images of the entire body of the driver are provided for the development of computer vision and machine learning-based algorithms.

This research study aims to improve road safety by applying efficient computer vision and machine learning based algorithms that can detect hazardous driving behaviours such as drowsiness and distraction. Computer vision approaches have been used for face/eye/mouth detection, image normalization, extraction of key features, etc. ML for classification of extracted features in conventional approaches and the learning, extraction and classification of features in the deep learning approaches. Initially, a comprehensive subject review of the most-related literature has been performed. Two different categories of approaches have been identified through the subject review and investigated in the practical work, the category of conventional approaches and the category of deep-learning approaches. The first phase of practical work has involved the use of conventional approaches for the drowsiness-related visual information analysis and predictions. The idea of using the combination of different drowsiness-related features such as eye blinking, yawning and head pose has been adopted to classify drowsy driving effectively.

For the second phase of the practical work, the focus was shifted towards studying deep learning approaches such as Convolutional Neural Networks (CNNs) for detecting distracted driving. In the literature, it has been identified that deep-learning approaches, for instance, CNN are type of deep networks explained in detail in Chapter 5, which involve the extraction and learning of features automatically from the number of hidden layers in the architecture which improves the overall generalization problem. Based on the comprehensive findings from the literature review, for the distraction detection, we have implemented four different pre-trained deep CNN architectures; AlexNet, ResNet, NasNet and MobileNet for detecting distraction among drivers. The proposed models were trained over the large dataset provided by Kaggle [17] and the performance was evaluated and compared.

1.2. Problem Statement

Research carried out in this thesis is within the scope of detecting hazardous driving behaviours such as the distraction and drowsiness using the computer vision and machine learning-based approaches. The monitoring of driving behaviours and the detection of hazardous driving behaviours can significantly improve road safety. Automobile manufacturing companies are investing and interested in developing driving-attention monitoring systems to improve road safety. These systems can warn drivers, most especially when their attention level (distraction and drowsiness) exceeds a certain threshold. Although in the literature there is extensive research in the domain of drowsiness and distraction available, it is rare to see these systems in practice. Furthermore, if the accuracy of is still not satisfactory and false warning of distracted driving is constantly issued, this can be intrusive and thus distract drivers attention unnecessarily and cause unnecessary negative impact to road safety. The objectives of this research are as follows:

- To identify the existing approaches and potential limitations in their practical applications.
- To detect facial regions relevant to drowsiness and distraction detections such as eyes, mouth, face and head positions.
- To extract drowsiness and distraction-related features from the detected facial regions such as the yawning rate, blinking rate, head nodding and head pose.
- To develop an efficient classification algorithm that can classify the extracted facial features and decide the level of distraction and drowsiness based on the classification results.
- To provide a detailed study of CNN and deep learning approaches for object detections (Deep learning approaches).
- To implement deep CNNs which can solve the Kaggle challenge for distracted drivers.

1.3. Research Questions

Base on the problem statement, the research questions explored in this research are as follows:

- **RQ1:** What are the potential challenges hindering the practical implementation of vision based drowsiness and distraction detection systems?
- **RQ2:** What conventional vision approaches can offer in detecting drowsiness and distraction in drivers?
- **RQ3:** How to measure and compare the performance of the conventional approaches? How is the performance of the conventional approaches in terms of detection accuracies and generalization?
- **RQ4:** What deep learning approaches can offer in the detection of distraction in drivers?
- **RQ5:** How to measure and compare the performance of the deep learning approaches? How is the performance in terms of their training, hyper-parameter time, classification accuracies, processing time, and memory consumption?

1.4. Thesis Layout

Rest of the chapters in the thesis are organized as follows. Chapter 2 presents the comprehensive subject review regarding the conventional and deep learning approaches for detection of hazardous behaviors in drivers. Chapter 3 presents the theoretical details of conventional vision approaches i.e. face detection, yawning detection, Viola and Jones, eyes detection. Chapter 4 presents the details regarding the implementation of conventional approaches and their corresponding results. Chapter 5 provides the theoretical background of deep learning approaches and different deep architectures used in this thesis. Chapter 6 presents the implementation of deep learning approaches and corresponding results and discussions. Finally, Chapter 7 concludes the thesis and provides potential future directions of research presented in this thesis.

CHAPTER 2

SUBJECT REVIEW

This chapter presents a subject review on the detection of hazardous driving behaviours (drowsiness and distraction) using the computer vision approaches; both conventional and deep learning. This chapter explores the different symptoms of drowsiness and distraction and their effects on the driving performance. Furthermore, this chapter reviews the latest literature regarding drowsiness and distraction detection in order to compare the performed research with the state of the art. Finally, this chapter highlights some potential challenges, identified from the literature, which have prevented the practical implementation of real-time driver monitoring systems.

2.1. Drowsiness Behaviors and Levels

Critical symptoms of drowsiness reported in literature include eye-blinking rate variations, a decline in driving concentration, a change in driver posture, steering grips, signs of depression, head nodding frequency, an increase in yawning frequency, a change in facial expressions, steering behaviour variations, confused thinking, reduced reaction responses, heart rate variations, skin potential variations, variations in brain signals, shallow breathing and frequency of touching face [18, 19]. Eye blinking rate, yawning rate and head position are the most significant signs of drowsiness in drivers.

Behaviours of drivers vary according to the level of drowsiness experienced by them. Based on research in [20, 21], Table 2.1 shows five different drowsiness levels and their corresponding behaviours, states and indicators.

Table 2.1: Drowsiness State Levels and Corresponding Behaviors, States and Indicators

Drowsiness Level	Behavior	State	Indicator
1	Reduced eye movement frequency and little opening of lips	Very Awake	Eyes widely open. very steady, thermal facial tone
2	Frequent movement of the eyes, motion is activated	Awake	Normal fast eye blinks; active eyeball movement; apparent focus on driving with occasional fast sideways glances; normal facial tone
3	Mouth movements, frequently touching on face and reseating	Drowsy	Increase in eye blinking duration, abrupt face rubbing, irregular eyes movement, restlessly seating and frequent yawning
4	Shakes head, frequent yawning, blinks are slow	Very Drowsy	Occasional disruption of eye focus, eye blinking duration increases, eyes openness decreases, reduced body movements and no facial tune for some
5	Eyes closed, dead fall (forward or backward)	Fatigue	Eyes completely closed, frequent yawning, complete disappearance of the facial tone

2.2. Types of Distraction and Their Causes

Distraction is a sub-type of in-attention, and the American Automobile Association Foundation for Traffic Safety (AAAFTS) as defined distraction as

"Slower response of drivers in recognizing the information needed to perform and complete successful/safe driving task because of some vehicles or outside vehicle events which shift the attention of drivers from driving task" [22]

According to [22], there are four main types of distraction: visual distraction, biomechanical distraction, cognitive distraction and auditory distraction. The definitions of each type of distraction is given below.

- **Visual distraction:** It is a type of distraction which involves the shift of drivers' visual field from driving by engaging in events such as observing in-vehicle objects or looking outside the vehicle [23].
- **Biomechanical distraction:** This involves the diversion of focus from driving because of the engagement of manipulating physical objects [24].
- **Cognitive distraction:** This type of distraction is directly related to thinking about other events while driving, a distraction that diverts the attention of drivers away from driving [25].
- **Auditory distraction:** It is a type of distraction which involves drivers' listening to audio devices such as radio and mobile phones while driving. Furthermore, this type of distraction may be due to drivers conversing with other passengers while driving [25].

Although the distraction is divided into four main categories, it has been observed that occurrence of distracted driving does not take place individually, Rather, the driver may encounter different types of distraction at the same time. In practical scenarios, all four types are inter-linked with each other and occur in combinations collaboratively. A perfect example of this interconnection is a driver answering a phone call while driving. In this particular case, all the four types of distraction mentioned above will occur. Visual distraction occurs when a driver looks at the display information on the cell phone before answering the call and locates the button to answer the call. Physical distraction occurs when a driver moves his hand from the steering to find the mobile phone in order to receive an incoming call. Cognitive distraction occurs when a driver in a call conversation shifts his/her thoughts toward the topic of conversation. Finally, auditory distraction occurs when a driver involves in conversation with someone on a call.

In a report published by the NHTSA [7], thirteen different sources of distraction were identified. These sources can be further categorized into three main streams: technology-based sources, non-technology-based sources and miscellaneous sources. Table 2.2 presents the categorization for the different sources of distraction.

Table 2.2: Different Sources of Distraction among Drivers Categorized by NHTSA [7].

Type of Distraction	Source
Technology-Based	Operating Radio and/or Music Devices Talking and/or Conversing on Mobile Phone Dialling and/or Using Mobile Phone Adjusting Climate Controls Using devices/objects brought into vehicles Using devices/controls integral to vehicles
Non-Technology-Based	Eating or Drinking Outside Object, Event or Person Other Passengers in Vehicles Moving Object in Vehicles Smoking
Miscellaneous	Other Distraction Sources Unknown Distraction Sources

Although the introduction of modern and state-of-the-art technological systems such as navigation and entertainment systems has facilitated drivers in many ways, they also contribute to distracted driving. This claim is well-supported by Stutts et al. [7] who predicted that the more the increase in advanced in-vehicle technologies, the more the chances of distraction-related accidents will rise. Stutts et al. [7] and Glaze and Ellis [26] investigated the impact of distraction and its contributions to road accidents. Stutts et al. explored the data from Crashworthiness Data System and highlighted the contribution of different distraction types in road accidents. On the other hand, Glaze and Ellis investigated the data from Troopers Crash Record and were focused on highlighting different sources of distraction and their involvement in road accidents. Based on these two studies and above-mentioned sources of distraction, a comparison has been carried out to study the impacts of distraction on road accidents and the involvement of different distraction sources. Table 2.3 lists the results.

Table 2.3: Contributions of Different Distraction Sources to Vehicle Crashes.

Distraction Type	Stutts et al. Study [7]		Glaze and Ellis's Study [26]	
	Distraction Sources	**% of Crashes**	**Distraction Source**	**% of Crashes**
Technology-Based	Adjusting radio, cassette, CD*	11.4	Adjusting radio, cassette, CD*	6.5
	Using/dialling mobile phone*	1.5	Using/dialling mobile phone*	3.9
	Adjusting vehicle/climate controls*	2.8	Adjusting vehicle/climate controls*	3.6
	–	–	Technology device*	0.3
	–	–	Pager*	0.1
	Total	**15.7**		**14.4**
Non-Technology-Based	Smoking related*	0.9	Smoking related*	2.1
	Other occupant in vehicle*	10.9	Passenger/children distraction*	8.7
	Eating or drinking*	1.7	Eating or drinking*	4.2
	Moving object ahead**	4.3	–	–
	Person, object or event**	29.4	–	–
	–	–	Grooming*	0.4
	–	–	Other personal items*	2.9
	–	–	Unrestrained pet*	0.6
	–	–	Document*	1.8
	Total	**47.2**		**20.7**
Miscellaneous	Other distraction	25.6	Other distraction inside vehicle*	26.3
	Unknown distraction	8.6	–	–
	Object brought in *	2.9	–	–
	Total	**37.1**		**26.3**

* Inside Vehicle Distraction Source

** Outside Vehicle Distraction Source

2.3. Stages of Drowsiness and Distraction Detection

In general, a drowsiness detection system consists of two main stages; first stage of extracting drowsiness related features from the facial information captured by sensor and second stage of classifying the extracted features to decide on the current state of driver (Drowsy or Active). Face, mouth, eyes and head pose are considered the most relevant facial features to detect the drowsiness in drivers. Figure 2.1 presents the basic structure of

a drowsiness detection system using computer vision technologies reported by Fuletra and Bosamiya [27].

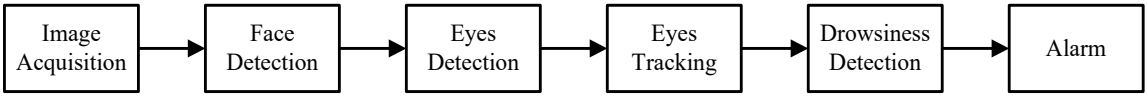


Figure 2.1: Basic Structure of Drowsiness Detection System using Computer Vision Based Techniques by Fuletra and Bosamiya [27].

Towards the distraction detection, eye glance of drivers has been considered one of the credible measure by researchers [28]. In eye glance approach, total time for eyes off the road is measured for the drivers when involved in performing some secondary task other than driving [29]. Head movements and eye glances of driver are captured and monitored using the camera sensor. Further, modern computer vision based tools such as FaceLAB [6] are used by the researchers in this domain to measure the eye glances based on eye tracking and head tracking information.

Visual occlusion is another commonly practiced approach by researchers towards detecting distraction in drivers. In visual occlusion technique, visual distraction of drivers is mimicked by temporarily blocking the view of drivers and to measure the road off the eyes. This approach considers that driver does not look on the road always but can be involved in short interval secondary tasks such as adjusting radio and controlling climate. All the secondary tasks which can be performed by driver within two second interval are classified as accepted tasks under this approach. For the defined occluded time interval, a driver can perform secondary tasks without visually looking at them which give the estimate of drivers visual demand to perform a task without getting distracted visually [30]. Visual occlusion based distraction detection approach is considered a promising approach by researchers [31-33].

2.4. Conventional Approaches for Drowsiness and Distraction Detection

This section presents the review of some benchmark existing approaches in the literature regarding the detection of drowsiness and distraction state/level among drivers using the

computer vision conventional approaches. Conventional approaches involve the extraction of features using the traditional vision-based approaches, the mathematical expressions or manually crafted approaches.

Gao et al. [34] proposed and developed a monitoring system to detect the emotional state of drivers in real-time scenario. The system was non-intrusive, and it used facial expression analysis to detect the emotional state. Facial expressions of drivers were captured in real-time by using in-vehicle camera sensors and were classified into two stress-related states: anger and disgust. The pose normalization approach was used to reduce the effect of head position on the detection and classification results. Figure 2.2 presents the block diagram of the system. The system in [34] included two modules: the face acquisition module and stress detection module working in sequential settings. In the face acquisition module, a Near-Infrared (NIR) was used to capture the real-time images of drivers, and facial landmarks were tracked using the face tracking system. In the stress detection module, at the first stage, the relevant facial features such as face and head pose were extracted using the local descriptors and holistic affine warping from the input captured data of the first module. Furthermore, before moving into the classification stage, pose normalization was applied. In the second stage, the module also provides a set of facial landmarks for the subsequent stress detection module. In the stress detection module, holistic or local texture features are extracted from the normalized facial images. They extracted features were classified using the Support Vector Machine (SVM) technique which is a space vector based machine learning classifier and aims to find the boundary between multiple output classes in a hyperplane, to decide the current emotional state of drivers. Based on the emotional state, the stress level of drivers was determined. SVM was trained offline over the pre-defined data. Two databases were used for this purpose, Radbound [35] and FACES [36], containing frontal view, evenly illuminated images from 49 and 179 subjects, respectively. In [34] to evaluate their system, two datasets containing images and videos were captured using the in-vehicle NIR camera. Dataset1 was recorded in the office configuration and included data from 21 different subjects. Dataset2 was recorded in the vehicle configuration and included data from 12 different subjects. The algorithm proposed by [34] was independently evaluated for both datasets, and an accuracy of 90.5% and 85%

was achieved for office and in-vehicle datasets, respectively. Although the algorithm exhibited promising performance over both evaluation datasets, it was not tested for different lighting conditions. In order to implement this system practically, it is of significant importance to validate the system over dataset containing images with different illumination conditions because it has been observed that the performance of vision based algorithms is sensitive to lighting conditions. Furthermore, the system only used the facial expression analysis to decide the stress level which is most probable to fail in situations where the face is not visible; thus, other approaches such as head movements and yawning can be integrated to further improve the results.

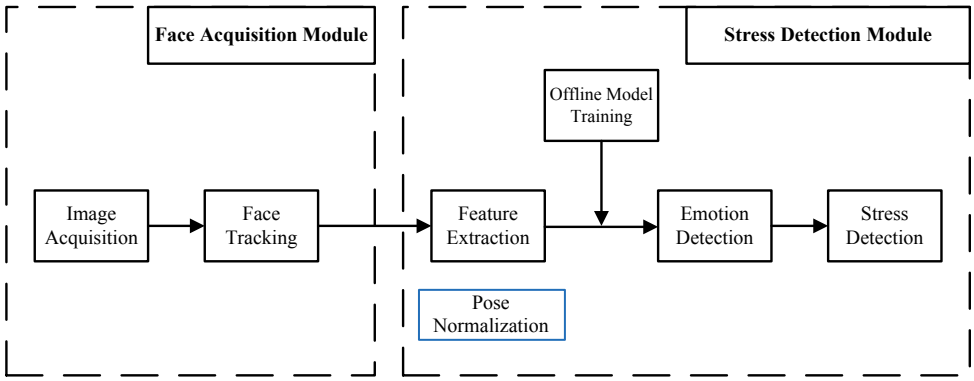
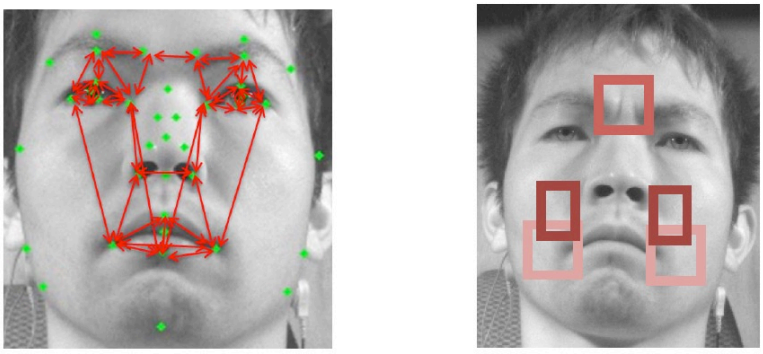


Figure 2.2: Block Diagram of Stress Detection System Proposed by Gao et al. [34].

Nakamura et al. [20] developed a system to detect the drowsiness among drivers by proposing the use of facial expression variations captured using infrared camera sensor. Authors proposed the idea of using variations in facial wrinkles along with the eye blink detection to make the system more robust and accurate. Variations in facial wrinkles were determined by calculating the local edge intensities of captured faces. Eye blinking was detected by calculating the distance between different feature points as demonstrated in Figure 2.3 (a). Textural variations were calculated by determining the local edge intensities at different facial regions such as mouth, eyebrows and nasolabial fold areas as demonstrated in Figure 2.3 (b). Laplacian filters were implemented using facial feature points. Extracted feature vectors of facial variations and local edge intensities were classified using a k-nearest neighbors algorithm (k-NN) based estimation algorithm to

determine the precise drowsiness state of drivers. From the results, the authors reported an improvement in detection by using textural features and feature point distances. The recommended algorithm was evaluated over the custom collected dataset from ten different subjects. An overall accuracy of 82.2% was reported for the proposed algorithms from the experimental evaluation. Although the proposed method of using wrinkle based textural features is the novel approach in drowsiness detection, it is important to implement it in real-time. Besides, more extensive validation of the proposed algorithm over the large datasets, including different illuminations, is needed.



(a) Distance parameters (b) Filtering area

Figure 2.3: Distance Parameters and Filtering Area for Eye Blink Detection and Textural Changes Proposed by Nakamura et al. [20].

Tadesse et al. [37] proposed a drowsiness detection system based on the Hidden Markov Model (HMM), a dynamic modelling of facial expressions. [37] uses HMM to perform temporal analysis of the dynamics of facial expressions. The detection of facial in each video frame was carried out by a system as shown in Figure 2.4. They developed a temporal analysis based system which included, change in facial features images caused by the facial movements and compared the proposed temporal analysis based drowsiness detection system with the frame based drowsiness detection system (see Figure 2.4) to highlight its advantages. The dynamic temporal analysis based drowsiness system utilizes the changes in the facial expressions of drivers and tries to associate the relation of those variations with the current drowsiness state of drivers. The authors used yawning, eye gaze, eyelid position and eye blinking as the facial expressions. The inclusion of additional facial features, including the eye blinking, improved the overall accuracy of the system. To detect and track

the face of drivers, Viola and Jones, and Camshift algorithms were applied to the input from the camera sensor. Important facial features from the captured faces were extracted using the Gabor Wavelet Descriptors. Extracted features were selected in two stages before the final classification stage. In the first stage, a simple threshold-based Adaboost weak learning algorithm, which is a popular learning algorithm used for image classification and face detection. The main idea of AdaBoost is to construct a succession of weak learners through different training sets with different weights. The training sets are derived from resampling the original data and the weights of the hard-to-leam instances will increase during every resample which show the main feature of the AdaBoost. These weak learners are fused through a weighted vote to predict the class label of a new testing instance. They are very effective on the initial stages in eliminating unwanted features. Usually the performance of a weak learner should be slightly better than random guessing and the weak learner is called as base classifier or component classifier. After boosting the final strong classifier can achieve high accuracy and good generalization ability. The authors applied this algorithm in the first stage. In the second stage, adaptive-boosting-based strong classifier was used to. Finally, the selected features from the second stage were classified using the SVM and HMM to decide the drowsiness state of drivers. The proposed algorithm was evaluated over the custom collected dataset from two driving subjects through simulated driving environments for different driving conditions. From the experimental evaluation of the system, an overall accuracy of 97% and 90% was achieved for the HMM-based classifier and SVM-based classifier. Although the HMM-based classifier demonstrated a promising detection accuracy, the proposed method was not validated for different illumination conditions and diversity of drivers which are important features towards real-time implementation of drowsiness detection systems.

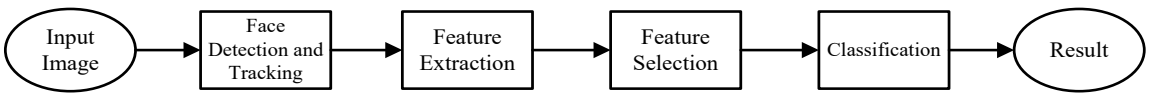


Figure 2.4: The Block Diagram of the System Used in [37] to Detect Facial Expression in a Single Video Frame.

D'Orazio et al. [38] suggested an attention monitoring system for drivers by using eye analyses and head movements. The team of authors proposed the monitoring of drivers' eyelids to check if it is opened or closed. Authors used two candidate regions that could contain eyes and used neural classifiers to recognize the eyes from input image sequences. Based on the openness of eyes, the current attentive state of drivers was decided. The eye closure duration and the frequency of eye closure were used as measures for behavioural analysis of determining the level of fatigue among drivers. The proposed algorithm was tested for its eye detection capability and behavioural analysis on a custom collected dataset. Datasets consisted of image sequences of drivers which were captured in the laboratory and in driving situations. An evaluation of the behavioural analysis was performed over eye closure parameters collected from image sequences of two different subjects. From the experimental evaluations, a maximum detection accuracy of 95% and 70% was achieved for subjects with open eyes and partially open eyes, respectively. The proposed algorithm could train itself adaptively during the driving and introduce the novel idea of reading driving habits of drivers to respond accordingly.

Saradadevi and Bajaj [39] developed a fatigue detection system for drivers by analysing the yawning information of drivers captured from the camera sensor. The Viola and Jones algorithm was used by authors to locate the face and mouth of drivers, and the SVM classifier was used to decide the fatigue level of drivers. The suggested algorithm was validated for a custom recorded dataset, and a performance of over 80% detection accuracy was achieved. Although the proposed system showed promising performance, it is more probable to fail when the mouth is not visible in the frame. Hence, the use of other visual information such as eyes and head movements will enhance the overall performance of the proposed system. Furthermore, the proposed system was not validated against extensive dataset; thus, it could not be implemented in real-time unless it was tested for large datasets containing images from the diversity of drivers under variable lighting conditions.

Singh and Papanikolopoulos [40] proposed a system to detect fatigue and distraction among drivers using the facial feature analysis. A colour camera was used by the authors to meticulously scan the driver face for relevant fatigue and distraction-related features such as eyes, mouth and head pose. Prominent skin pixel variations were captured using the skin

colour mode. Skin like pixels from the input colour facial images were filtered using the skin colour model and were further processed using the blob processing approach to determine the connected areas and exact position of the face accordingly. To localize the eyes in the facial region, a horizontal gradient technique was implemented; however, the grey scale processing approach was used for real-time pupil detection. Information from this processing helped the system to capture the eye blinking variations and face directions. The proposed system was evaluated over a custom recorded dataset of drivers with different skin colours, gender and facial hair. From the experimental results, authors achieved a prolonged eye blink detection accuracy of approximately 95%. Although the proposed system exhibited promising results for the fatigue detection among drivers, it was not tested for different illumination conditions. Furthermore, other fatigue and distraction-related facial features such as yawning, expressions, and nostrils can be integrated to enhance the performance of the overall system.

Percentage Eye Closure (PERCLOS) [41] is one of the most commonly used and reliable measures for detecting visual distraction reported in the literature. PERCLOS is the percentage of eye closure over a given time interval. The feature of detecting slow eye closure rather than eye-blinking makes PERCLOS an accurate measure to detect eye-blinking among drivers [41]. Bergasa et al. [42] proposed an attention monitoring system by detecting the level of drowsiness and distraction among drivers. The authors used the head position information to determine the level of distraction, and PERCLOS and yawning rate to measure the drowsiness level of driver. The combination of all three factors helped the system to detect the attention level of drivers efficiently. They also used RANSAC and POSIT 3D face tracking models to estimate the head movements. User-and-illumination-independent model for facial expression was used to detect different facial features. Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) were applied to reduce the input data dimensions. PERCLOS threshold was used to decide the attention level of drivers. The proposed algorithm was evaluated over a custom recorded video dataset in simulating the environment and actual driving scenario. Cohn-Kanadae dataset [43] was used to train the facial expression classifier mode. Facial expressions were efficiently detected by the proposed algorithm for the given dataset and were not

significantly affected by illumination conditions: however, the authors did not report any numerical results. The face pose estimation algorithm was able to track the face with $\pm 40^{o}$ face rotation accurately. Based on the performance results of two subsystems, the authors achieved an accuracy of 92% in detecting the attention state of drivers.

Dasgupta et al. [44] proposed a driver attention monitoring system based on PERCLOS to determine the level of drivers attention. They used Kalman filtering and Haar-like features to track and detect the face captured by the camera sensor used in detection of eyes at day and night using feature techniques. The authors proposed two different methods for different lighting conditions to detect and extract the eyes from detected faces. In order to extract more effective features from the static images, Local Binary Patterns (LBP) features and the principal component analysis (PCA) approach are used. The PCA is used to reduce dimensions of the features which are combined by the gray pixel value and Local Binary Patterns (LBP) features is used as an important descriptor for the pattern analysis of image, the authors used these feature technique to get the texture information from the images. All the features are extracted from the active facial patches. The active facial patches are these face regions which undergo a major change during different lighting conditions. PERCLOS values were classified to determine the closeness and openness of eyes using the SVM classifier, and, accordingly, the attention level of drivers was determined. To compensate the face rotations during real-time driving scenarios, affine and perspective transformations were applied. Furthermore, the Bi-Histogram Equalization (BHE) approach was used to compensate for the different illumination conditions in real-time driving situations. The proposed algorithm for attention monitoring of drivers was evaluated for simulated driving conditions and actual driving conditions in both daylight and night scenarios. The scholars achieved an overall classification accuracy of 98.6% with 9.5 framers per second speed.

Ji et al. [45] recommended a fatigue monitoring system for drivers in real-time situations based on IR illumination device and Charge-Coupled Device (CCD) camera. The idea of IR illumination devices was used by the authors to improve the monitoring results in low light conditions. Combination of different behavioural measures such as gaze estimation, eyelid movement, facial expressions and head pose was used to determine the driver alertness. Eyes detection was achieved by using SVM classifier while for eye tracking a

combined approach of using Kalman filters and mean shift tracking was used. Talking more technically, PERCLOS and Average Eye Closure Speed (AECS) [46] were used as a measure to decide the driver alertness. A probabilistic model (Bayesian Networks) was established by the authors to mimic human fatigue and visual information was used to predict the current fatigue level of drivers. The proposed system was evaluated in two part by authors. In the first part, detection accuracies of individual computer vision based facial feature detection algorithms was validated while in the second part, fatigue parameters validity was evaluated. For eye detection, a huge dataset of 13620 images was used and detection accuracy of 95.8% was achieved by the researchers. For head pose, an estimated root mean square error of 1.92 degrees and 1.97 degrees was achieved for pan and tilt angles, respectively. For fatigue parameters evaluation, data from eight subjects was collected and response time was used as metric for performance. However, no numerical results were reported by the authors in this regard.

Sacco et al. [47] developed a driver alertness system in real-time by utilizing facial features of drivers. Important visual facial features such as the face, eyes and mouth from the camera input were detected using the Viola and Jones algorithm. Real-time tracking of the detected facial features was achieved by using the template matching approach. Extracted facial features were then classified using the SVM classifier. PERCLOS, average eye closure interval and degree of mouth openness were utilised in deciding the level of alertness among drivers. Viola and Jones face, eye and mouth detection algorithms were evaluated over 6,000 images dataset containing the exact half of negative and positive images. FERET dataset was used for the positive images, while negative images were captured from an in-vehicle video feed. As for the face detection, an accuracy of 99.9% was achieved, whereas as for the eye detection and mouth detection, detection accuracies of 96% and 91.9% were achieved. The overall accuracy of 93.24% and 95.20% were achieved by authors for SVM with linear kernel and SVM with Radial Basis Function (RBF) kernel, respectively.

Bergasa et al. [48] proposed a real-time non-intrusive driver vigilance monitoring system using the computer vision approach. The authors used the IR illuminated camera sensor hardware to capture the facial information of drivers. Six different vigilance related

parameters were calculated in the proposed research: the PERCLOS, eye closure interval duration, frequency of eye blinking, frequency of head nodding, position of face and fixed gaze directions. Computer parameters were combined into fuzzy-based classifier, and decision on the driver attentiveness was made. The suggested algorithm was evaluated over the custom collected dataset of real-driving from ten different users in both day and night lighting conditions. The authors achieved the detection accuracies of 93.12%, 84.37%, 80%, 72.5%, 87.5% and 95.2% for PERCLOS, eye closure duration, eye blinking frequency, head nosing frequency, face position and fixed gaze direction, respectively.

Fan et al. [49] suggested a yawning detection system for measuring the fatigue among drivers. They used CCD camera sensor to capture the facial and mouth information of drivers. Face detection was achieved using the Gravity-Center template approach, while mouth corners were detected with the grey projection approach. Furthermore, the authors used the Gabor Wavelets approach to extract the texture related features from the mouth. Extracted yawning related features were then classified using the LDA classifier, and the decision on yawning was made. They also evaluated the proposed algorithm over 400 image sequences selected from twenty different video sequences. From the experimental results, the authors achieved an overall yawning detection accuracy of 95%.

Vural et al. [50] proposed a drowsiness detection system using machine learning on facial movements of drivers. Machine learning approach was used by the authors to determine the actual behaviours of drivers during drowsiness episodes. Machine learning classifiers were developed using 30 different facial actions from Facial Action Coding. Classifier included drowsiness-related facial actions such as eye blinking, yawning and head movements. Information of these facial actions was passed to Adaboost and multi-nomial ridge regressor to predict the sleep and crash episodes among drivers. Proposed algorithm was evaluated using a driving computer game simulation and classification accuracy of 96% and 90% was achieved by the authors for in-subject and across-subject, respectively.

Yin et al. [51] proposed a novel approach for fatigue detection among drivers based on the multiscale dynamic features of facial images. Multiscale representations from input image sequences were achieved by using Gabor filtering technique. At the next stage, from each

multiscale image, LBP were extracted. Resulted LBP of image sequences were divided into dynamic features and were concatenated with the histogram of each dynamic feature. A statistical based learning algorithm was used to select the most distinguished dynamic features and at the final stage a strong classifier was used to classify the state the drowsiness among drivers based on selected dynamic features. The proposed algorithm was evaluated over custom collected data set of 600 images coming from thirty different subjects. From the experimental results, the researchers achieved the classification accuracy of 98.33%.

Flores et al. [52] proposed a computer vision and artificial intelligence based Advanced Driver Assistance System (ADAS) to automatically detect the drowsiness among drivers. They used the visual information such as face, eyes and head position to decide the drowsiness and distraction level of drivers. Viola and Jones algorithm was used to detect the face while eyes were located by defining the regions of interest in detected faces. To track the eyes and face, fusion of condensation algorithm and neural networks was applied. Finally, eye state detection was achieved by using SVM classifier. Important features of eye were extracted using the Gabor filters. Drowsiness index was determined based on the PERCLOS measurements. The proposed algorithm was evaluated over the custom collected dataset under different lighting conditions and from the diversity of drivers. From the experimental results, on average accuracy of 90% was achieved.

Liu et al. [53] proposed a drowsiness detection system based on the eyelid movements of drivers. They used the cascade classifiers for the detection of the face from input sequences and diamond search algorithm to track the face in real-time. Temporal difference image approach was used to detect the eyelid movements. To judge on the performance of the proposed system, the authors used eyelid closure duration, group of continuous blinks and eye blinking frequency as three measures. The suggested algorithm was evaluated over custom collected low resolution dataset of ten different subjects. From the experimental results, an accuracy of about 98% was achieved by the scholars.

Zhang and Zhang [54] proposed a fatigue detection system for drivers based on the non-linear Kalman filters and eye tracking. Unscented transformations were used by the authors to non-linearly track the eyes in real-time. PERCLOS was used as a measure to detect the

fatigue level among drivers. Face detection was achieved using the SVM based classifiers. The proposed algorithm was validated over 20 qualified drivers with different genders and different ages. From the experimental results, eye detection accuracy of over 99% was achieved by the authors.

Park and Trivedi [55] proposed a head posed detection system towards determining the attentiveness of drivers using Support Vector Regressor (SVR). They implemented the universal motion approach and colour statistical orders to monitor and track facial behaviours of drivers. The proposed method was reported to be failed if driver is rotating its head in certain direction, conversing with other in-vehicle passengers and if wearing the glasses. Proposed algorithm was evaluated over a custom recorded NTSC image dataset of subjects performing different driving related tasks. The authors did not report any extensive evaluation results in numerical rather presented the results as segmented images. Improvements in the proposed system can be made by integrating other sensors such as thermal images and steering images. Furthermore, the fusion of this approach with vision based attention monitoring approaches will enhance the overall performance.

Nguyen et al. [56] proposed a computer vision based system to monitor the real-time drowsiness among drivers using the facial features of driver. A camera sensor was used to capture the RGB visual information of driver. For face detection, Haar features based approach was applied while for eyes detection random forest approach was used. From the detected eyes, binary images of local eye regions were extracted. From the binary images, a decision on drowsiness was taken based on closeness and openness of eyes. A pre-defined threshold was used to determine if the driver is drowsy or not. Proposed algorithm was evaluated over data collected from four different subjects in normal illumination conditions. From the experimental results, the detection accuracy of 97.6% and 94% was achieved for eye detection and eye-state prediction, respectively.

Kholerdi et al. [57] proposed a human visual system-based image processing approach to detect the drowsy behaviour among drivers. Images captured by the camera were pre-processed for illumination conditions, including the noise before getting into the feature extraction stage. The luminance variation model was used for illumination variation, while

spatio-temporal filters were applied to compensate for the noise. To detect the eyes and mouth from pre-processed images, the Viola and Jones algorithm was applied. Motion events around the eyes and mouth were captured by using energy of the Mango approach in the Region of Interest (RI). Finally, three measures – the head dropping, yawning and closed eyes –were used to define thresholds and conditions and decide the drowsiness level of drivers. The proposed system was evaluated for the custom defined dataset, and an overall classification accuracy of 90% was achieved by the authors.

Jo et al. [58] defined a term PERLOOK as a measure that detects the distraction level of drivers. A similar idea PERCLOS for drowsiness detection was used to define PERLOOK. They defined PERLOOK as the percentage of time interval which a driver is not looking straight i.e. a rotated head or eyes of the road. A certain threshold of PERLOOK was defined for active driving. If the detected PERLOOK value of drivers was greater than the defined threshold, the driver was classified as visually distracted. They used the yawning information to determine PERLOOK, and the eye blinking information to determine PERCLOS values of drivers. Then, they decided on the drowsiness and distraction level based on these values. PCA and LDA approaches were used to extract the features. Nabo [59] used a software toll, SmartEye [60], to determine the value of PERLOOK from camera input towards detecting the distraction. Extracted features were then classified using the SVM with BF kernel to determine the attention level of drivers. To evaluate the proposed system, a dataset of 162,772 images from 22 different subjects was collected. The proposed system was evaluated for eye detection, eye state detection and inattention classification. The authors achieved the detection accuracy of 98.58%, 98.55% and 97.09% for eye detection, open eye state detection and closed eye state detection, respectively. For attention level measurements, 0% error was achieved for recognizing inattentive state as the normal state, while 2% error was recorded for recognizing normal state as inattentive state.

Craye and Karray [61] proposed distraction detection and type recognition system for drivers using computer vision based approach. Overall, the proposed system includes four modules, eye behaviour module, arm position module, head pose module and facial features module. Distraction related features such as eye blinking/gaze direction, arm position, head position and facial expressions from each module were extracted, respectively. Extracted

features were then classified using two different classifiers, AdaBoost and HMM to decide the distraction state of driver. The proposed algorithm was evaluated over the custom collected video set and manually labelled for different distraction tasks. The authors achieved an accuracy of 85% and 84% for AdaBoost and HMM classification, respectively.

Liao et al. [62] proposed a novel approach for detecting the real-time cognitive distraction among drivers using SVM. The proposed system consisted of three main elements: a feature calculation module, SVM classification module and a filtering recognizer module. The algorithm used the facial information like gaze direction and head position from facial images while incorporating the steering angles and speed of vehicles from in-vehicle sensors. From the collected information, distraction-related features were calculated and extracted. Extracted feature vectors were initially classified by SVM and were stored in the buffer. Finally, a filtering recognizer was used to classify the attention level of drivers. To validate the final classification, authors implemented a consistency tester. To evaluate the proposed algorithm, they collected data from 26 different subjects using a driving simulator under both urban area conditions and highway conditions. Froom the experimental results, the researchers achieved the classification accuracy of 93% and 98.5% for highway and urban areas, respectively.

Kircher et al. [63] proposed two algorithms, Percentage Road Centre (PRC) and the 3D model of vehicles to detect the distraction among drivers. The first algorithm used the gaze direction information of drivers, and the authors decided the type and level of distraction based on the PRC measured values. PRC greater than 85 was classified as cognitive distraction sand less than 58 was decided as visual distractions. In the second algorithm, they defined the visual field of drivers by utilizing the internal 3D model of vehicles. A driving-related visual field was defined by the authors; if the detected eye glance of drivers was within the defined visual field, they were classified as active otherwise distracted. For the evaluation purposes, seven subjects (4 males and 3 females) were involved in the study. However, the authors did not report any numerical results; rather, they presented only the qualitative analysis.

Pohl et al. [64] recommended a driver distraction detection system by using gaze direction and head pose as measures of detection. They used a decision maker to classify the distraction level among drivers accurately. Furthermore, lane keeping modules and lane position modules were used by authors to detect distraction. The proposed system was evaluated by the authors for true positive, false positive, and true negative interventions.

Murphy-Chutorian et al. [65] suggested a distraction detection system based on the head pose detection. They used the localized gradient orientation histogram approach to extract the distraction-related features. The authors also classified those features using SVM regression to decide the level of distraction. Head orientation was estimated in two degrees of freedom (yaw and pitch). They used an experimental testbed to mimic real driving situation, and data were collected to evaluate the proposed algorithm. Data from ten different subjects were captured and used to train the head pose estimation classifier. The mean absolute error was used as a measure of performance for the proposed head pose estimation algorithm. For laboratory experiments, an error of 5.58 and 6.40 degree was recorded for yaw and pitch angles. For daylight driving experiments, an error of 3.99 and 9.28 degree was observed for yaw and pitch angles, respectively. Finally, for nightlight driving experiments, an error of 5.18 and 7.74 was achieved for yaw and pitch angles, respectively.

Table 2.4 presents the summary of overall cited literature related to drowsiness and distraction detection using conventional approaches. Table 2.5 presents the categorization of cited literature in terms of what detection measure being used, what approach was for detection used and what classifier implemented to achieve the final decision.

Comprehensive literature study and critical analysis have been reported regarding the use of conventional computer vision approaches in detecting drowsiness and distraction in drivers. Mostly, the approaches involve the crafting of facial features related to drowsiness and distraction manually and classifying manually crafter features using pre-trained machine learning classifiers. However, from the literature it has been identified that conventional approaches are poor in handling the generalized situations and hence are not implemented in real-world applications. As part of this research, conventional approaches

based eye detection, yawning detection and head pose detection systems have been implemented to detect the drowsiness in drivers and are critically analysed. Chapter 3 and Chapter 4 of this thesis presents the theoretical and implementation details of conventional computer vision approaches.

Table 2.4A: Comparison of Literature Related to Drowsiness and Distraction Detection using Conventional Computer Vision Approaches.

Authors	Database Information	Features/Measures	Facial Elements	Methods/Techniques	Classifiers	Performance
Gao et al [34]	FACES and Radbound Dataset for training. Custom Dataset for evaluation	Eye blinking, Yawning and Head pose Anger and Disgust stress related emotional states	Face, Eyes, Mouth and Head	Holistic affine, wrapping, local descriptors, Gabor Wavelets	SVM, Ratio of eye-height and eye-width	Office datateset:90.5% In-vehicle dataset: 85%
Nakamura et al. [20]	Custom dataset collected from 10 subjects	Eye blinking, Changes in Wrinkles	Face and Eyes	Laplacian filters, edge detectors in addition with feature methods	K-NN Neural Classifier	82.2%
Tadesse, et al. [37]	Custom dataset collected from 2 subjects	Eye blinking, eye gaze, head pose and yawning	Eyes, mouth, skin, face and lips	Gabor wavelet decomposition Adaboost	SVM and HMM	HMM: 97% SVM: 90%
D'Orazio, et al [38]	Custom dataset collected from 2 subjects	Eyelid movements and head pose Eye closure duration and eye closure frequency	Eyes, face and head	Hough Transforms and Neural Netowrks	Normal Behavior Model	Open eyes: 95% Partially open eyes: 70%
Saradadevi and Bajaj [39]	Custom Dataset	Yawning	Face and mouth	Viola and Jones algorithm	SVM	80%
Singh and Papanikolopoulos [40]	Custom Dataset	Skin variation, eye blinking, face direction	Face and Eyes	Skin Color Model, Horizontal Gradient and Grayscale Processing	Eye Closure Duration Thresholding	Eye Blink Accuracy: 95%
Bergasa et al. [42]	Cohn-Kanadae Dataset	Head pose, eye blinking and yawning PERCLOS	Face, eyes, mouth and head	RANSAC, POSIT 3D, PCA and LDA	PERCLOS Threshold	Head Pose: $\pm 40^{o}$ Attention Detection: 92%
Dasgupta et al. [44]	Custom Dataset	Eyelid movements PERCLOS	Face and eyes	Kalman Filters, Haar-Like Features, PCA, LBP and BHE	SVM	98.6%
Ji et al. [45]	Custom collected dataset of 13620 images	Gaze estimation, eyelid movements, facial expressions and head pose PERCLOS and AECS	Face, eyes and head	Kalman Filters and Mean Shift Tracking	SCM and Bayesian Networks	Eye detection: 95.8% **Mean Square Error** Pan Angle: 1.92 Tilt Angle 1.97
Saccor et al. [47]	FERET Dataset	Eye closures, degree of mouth openness PERCLOS	Face, Eyes and Mouth	Viola and Jones, and Template Matching	SVM	Face detection: 99.9% Eye Detection: 96% Mouth Detection: 91.9%

Table 2.4B: Comparison of Literature Related to Drowsiness and Distraction Detection using Conventional Computer Vision Approaches.

Authors	Database Information	Features/Measures	Facial Elements	Methods/Techniques	Classifiers	Performance
Bergasa et al. [48]	Custom Dataset	Eye blinking, head nodding, gaze direction and face position PERCLOS, eye closure	Face, Eyes and Head	Finite State Machine, Blob Detection and Kalman Filters	Fuzzy Classifier	PERCLOS: 93.12% Eye Closure Duration: 84.37% Eye Blinking Frequency: 80% Head Nodding: 72.5% Face Position: 87.5% Gaze Direction: 95.2%
Fan et al. [49]	400 images from 20 videos	Yawning	Face and Mouth	Gravity-Center Templates, Grey Projection and Gabor Wavelets	LDA	95%
Vural et al. [50]	Driving Game Simulation collected data	Eye blinking, yawning and head movements	Face, Eyes and Head	Machine Learning Approach	Adaboost and Multi-Nomial Ridge Regressor	Within Subjects: 96% Across Subjects: 90%
Yin et al. [51]	Custom collected dataset of 600 images from 30 subjects	Multiscale Dynamic Facial Features	Face	LBP and Statistical Learning	Cascade Strong Classifier	98.33%
Flores et al. [52]	Custom Dataset	Eye blinking and Head Pose PERCLOS	Face and Eyes	Viola and Jones, Condensation Algorithm, Neural Networks and Gabor Wavelets	SVM and PERCLOS Threshold	90%
Liu et al. [53]	Custom collected data from 10 subjects	Eyelid movements, eye closure and eye blinking	Face and eyes	Diamond Search Algorithm and Temporal Difference Images	Threshold based classification	98%
Zhang and Zhang [54]	Custom collected data from 20 qualified drivers	Eye closure and monitoring PERCLOS	Face and Eyes	Kalman Filters, Unscented Transformations	SVM	99%
Park and Trivedi [55]	Custom collected NTSC images	Facial behaviors and head pose	Face and Head	Universal Motion Approach and Color Statistical Model	SVM Regrtessor	No Quantitative Results Reported
Nguyen et al. [56]	Custom collected dataset from 4 subjects	Eye blinking and eye closure	Face and Eyes	Haar Features and Random Forest	Pre-defined Thresholds	Eye Detection: 97.6% Eye State Prediction: 94%
Kholerdi et al. [57]	Custom Dataset	Head dropping, yawning and eye blinking	Face, Mouth, Eyes and Head	Viola and Jones, Spatio-Temporal Filers and Energy of Mango Approach	Pre-defined Thresholds	90%

Table 2.4C: Comparison of Literature Related to Drowsiness and Distraction Detection using Conventional Computer Vision Approaches.

Authors	Database Information	Features/Measures	Facial Elements	Methods/Techniques	Classifiers	Performance
Jo et al. [58]	Data set of 162,772 images from 22 different subjects	Yawning and eye blinking PERLOOK and PERCLOS	Face, Mouth and Eyes	PCA+LDA	SVM with BF Kernel	Eye Detection: 98.58% Open Eye State: 98.55% Close Eye State: 97.09% Attention Level: 98%
Craye and Karray [61]	Custom Database	Eye blinking, gaze direction, arm position and head pose	Eyes, Head and Arm position	Marching Squares Algorithm and Hough Transform	AdaBoost and HMM	AdaBoost: 85% HMM: 84%
Liao et al. [62]	Data collected from 26 different subjects in urban and highway driving conditions	Gaze direction, head pose, steering angles and vehicle speed	Head	SmartEye software tool	SVM	Highway driving: 93% Urban driving: 98.5%
Kricher et al. [63]	Data from 7 subjects (4 Male and 3 Female)	Gaze direction and Visual field PRC and AttenD	Head and Eyes	Not Reported	PRC and visual field thresholds	No Quantitative Results Reported
Pohl et al. [64]	Custom Dataset	Gaze direction and head pose	Eyes and Head	Not Reported	Decision Maker	No Quantitative Results Reported
Murphy-Choutrian et al. [65]	Date collected from different subjects using experimental testbed	Head pose and orientation	Head	Local Gradient Orientation Histogram	SVM Regression	**Mean Absolute Errors** **Lab Experiments** Yaw Angle: 5.58 degree Pitch Angle: 6.49 degree **Daylight Experiments** Yaw Angle: 3.99 degree Pitch Angle: 9.28 degree **Night Experiments** Yaw Angle: 5.18 degree Pitch Angle: 7.74 degree

Table 2.5: Categorization of Conventional Computer Vision Approaches from Literature for Distraction and Drowsiness Detection.

	Detection Measures					**Approaches to Feature Extraction**		**Classifiers**				
	Eye blink	Yawning	Head pose	Gaze direction	Other	Handcrafted Filtering	Machine Learning	Threshold	Fuzzy Logic	LDA	SVM	Other
Drowsiness	[20] [34] [37] [38] [40] [42] [44] [45] [47] [48] [50] [52] [53] [54] [56] [57] [58] [61]	[34] [37] [39] [42] [47] [48] [49] [50] [57] [58]	–	–	[20] (change in wrinkles) [34] (Emotional states) [40] (Skin variation) [51] (Multiscale facial features) [55] (Facial behavior)	[20] [34] [37] [38] [40] [42] [44] [45] [48] [49] [53] [54] [55] [57] [58] [61]	[38] [39] [44] [47] [50] [51] [52] [56] [57]	[34] [40] [42] [52] [53] [56] [57]	[48]	[42] [49]	[34] [37] [39] [44] [47] [52] [54] [55] [58]	[20] (KNN) [37] [61] (HMM) [38] (Normal Behavior Model) [45] (SCM and Bayesian) [50] (Ridge Regressor) [51] (Strong Cascade)
Distraction	–	–	[34] [37] [38] [42] [45] [48] [50] [52] [55] [57] [58] [61] [62] [64] [65]	[37] [45] [48] [61] [62] [63] [64]	[61] (arm position) [62] (in-vehicle) [63] (visual field PRC)	[34] [37] [38] [42] [45] [48] [55] [57] [58] [61] [65]	[38] [50] [52] [57]	[42] [52] [57] [63] [64]	[48]	[42]	[34] [37] [52] [55] [58] [62] [65]	[37] [61] (HMM) [38] (Normal Behavior Model) [45] (SCM and Bayesian) [50] (Ridge Regressor)

2.5. Deep Learning

Artificial intelligence, machine learning and deep learning are often associated together and interlinked. Figure 2.5 presents the relation between artificial intelligence, machine learning and deep learning.

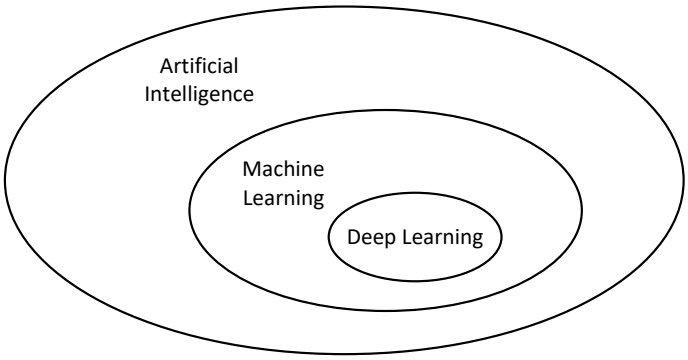


Figure 2.5: Relation between Artificial Intelligence, Machine Learning and Deep Learning [66].

Artificial intelligence is a broad domain of computer science which explores the concept of making computing machines to be able to think and solve problems intelligently like humans. Artificial intelligence involves various approaches to achieving the defined task, and machine learning is one of the fields which can achieve this goal. Machine learning, on the other hand, revolves around the concept of teaching a computer to solve a certain problem by training it with sample data. This process is also referred to as learning representation through data in technical language. It is important to understand how learning takes place in this process. The process of learning involves the meaningful transformations of input data automatically. These meaningful transformations are also referred to as the learned feature representations, which are achieved using different functions. Conventional machine learning algorithms involve the role of the engineer to define human-crafted features or what type of features needs to be extracted. This makes the conventional machine learning algorithms subjective in nature; that is why they differ in results significantly when applied to different situations and when they are trained by

different human-crafted features. In contrast, deep learning automates the feature selection and extraction process [66].

Deep learning is a branch of machine learning which uses deep layered network architectures to learn the feature representations in a hierarchal way i.e. learning of low-level features at initial layers of the network while higher level features towards the end of the network. Deep learning provides the advantage of learning feature representations automatically without any human intervention, an advantage which was the major drawback of conventional machine learning algorithms. Deep architectures are basically inspired from the animal cortex system which consists of a number of hidden layers unlike the shallow architectures which have only a few hidden layers [67]. Each layer in the architecture transforms the input data which is characterized by the weights of the layer. The goal of deep learning is to train the layer weights so that inputs are mapped to the output with minimum possible losses. This objective seems quite simple, but in practice, deep architecture consists of a number of layers and involves the learning of a large number of parameters. Given that changing the value of one parameter will change the behaviour of all the other parameters, learning large number of parameters is a relatively complex problem. To measure the performance of predictions made by a deep architecture, a loss function also known as the objective function is used to compare the predicted values with the ground truth and to generate a loss score. This loss score is then propagated backwards using some optimiser to update the weights of layers so that the loss score is minimized. The optimiser is basically the central element in the training of a deep architecture, and it usually implements efficient backpropagation algorithms such as the gradient descent. Figure 2.6 presents the basic functional diagram of how a deep architecture works.

At the beginning of the training process, the layer weights are initialized with random values in most cases. This means that some random transformations are applied to the input, thus making the predictions not close to the ground-truth (and hence a high loss score). However, as the numbers of iterations are performed, the weights are updated to adopt the minimization of the loss score to produce a better output. It is important to mention that the initialization of weights in a deep architecture in itself is an active area of research. With

the introduction of a Deep Belief Network (DBN) by Hinton et al. [68] in 2006, the training process of deep architectures was revolutionized.

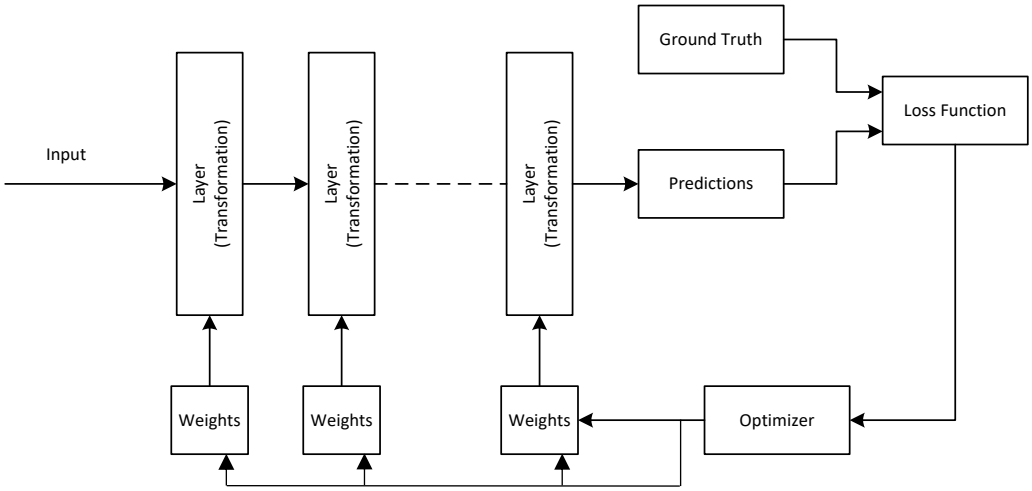


Figure 2.6: Functional Working Diagram of Deep Learning [66].

Deep learning approaches offer a number of advantages over the conventional machine learning methods, including better performances for a number of tasks and effective computational abilities. However, the most important advantage of deep learning is the automation of feature engineering processes for complex real-world tasks in improving performances, efforts and generalization. Despite all the advantages, the efficiency of deep learning is encumbered with the following limitations.

(a) Overfitting: Given that deep architectures involve a large number of parameters at different layers, the problem of overfitting is likely to be encountered during the training process.

(b) Training Data: Deep architectures require a huge amount of data to train the model efficiently and to achieve better performance. However, the availability of such amount of data is not always possible. To deal with the problem of data insufficiency, researchers now use augmented datasets [69, 70] and transfer the learning of pre-trained models [71, 72].

(c) Computational Resources: To train a deep architecture, huge amount of computational power is needed depending upon the complexity of the architecture

and the nature of the problem being solved. However, the computational resources are only needed at the time of training. Once the model is trained, it can be used in real-time applications using the existing hardware commercially. Given the availability of Graphical Processing Units (GPUs) in the market and the concept of cloud computing, training deep architectures is no longer considered a limitation. Nevertheless, such training is time-consuming.

(d) Parameter Optimization: Although deep architectures can be used to solve complex problems successfully, a number of questions remain unanswered, for example which model is appropriate for a given problem, why the performance of one model is better in one situation than the other, and how to optimise different parameters involved in the deep architecture.

2.6. The Architecture of Convolution Neural Networks

CNNs also known as ConvNet are a class of deep-learning feed-forward networks widely used for processing grid-like topology data i.e. images. CNN models are designed to emulate the visual cortex behaviour and are considered the most powerful models that can perform computer vision tasks such as image classification. The CNN architecture consists of convolution layers, pooling or down-sampling layers, non-linear layers (activation functions), fully connected layers, classifiers, loss functions and optimizations [73].

2.6.1. Convolution Layer

A convolution layer is the principal operation in these networks whose task is to transform the input data to extract features without losing the spatial dimensions/information of the input data. Equation 2.1 presents the mathematical expression for the convolution operation.

$$F^H = \sum_{i=1}^{q} \mathbf{W}_i * \mathbf{x}^i + b \tag{2.1}$$

where $\mathbf{x}$ is the input image of convolution layer with the dimension $N \times N \times D$, $\mathbf{W}$ is the filer of size $k \times k$ with H kernel size and F^H is the convolved output with $k \times k \times H$ size.

At different convolutional layers, different filters are used. Each filter extracts different feature representations. In practice, it is common to use multiple filters at each convolutional layer. The output of the convolutional layer is usually referred to as the activation maps which are then fed to an non-linear activation function. The size of convolutional layers is dependent on three factors: the size of stride or filter translational jump, the size of padding to the input image, and the number of filters used at the layer (depth).

2.6.2. Pooling Layer

A pooling layer is also referred to as the down-sampling layer and usually follows the convolutional layer periodically. The primary objective of a pooling layer is to reduce the spatial size of activation maps, the computations and the size of the number of parameters to avoid overfitting problems. The most commonly used pooling approach is max pooling in which input maps are divided into clusters, commonly a 2 by 2 rectangle with a stride of 2, and a maximum value of each cluster is taken. It is important to note that the depth of the data remains unchanged in pooling even though the spatial dimensions are reduced. Other pooling approaches available include average pooling and L2-norm pooling; however, in practice mostly max-pooling is used.

2.6.3. Fully Connected Layer

A fully connected layer is the final layer before classification. In this layer, all the learned classification maps are represented as vectors and are connected to all the neurons of the previous layer. These learned feature representations are then classified or predicted as one of the multiple output classes in classification tasks.

Deep learning has been effectively implemented by researchers in computer vision for visual-computation and face-recognition tasks. Some CNN-based deep learning architectures with validated performances on large datasets in computer vision include AlexNet architecture [74], ResNet50 architecture [75], NASNet architecture [76], and MobileNet architecture [77]. Some applications of deep learning include SLAM, natural language processing, deep reinforcement learning and semantic segmentation. Figure 2.7 presents a typical CNN architecture.

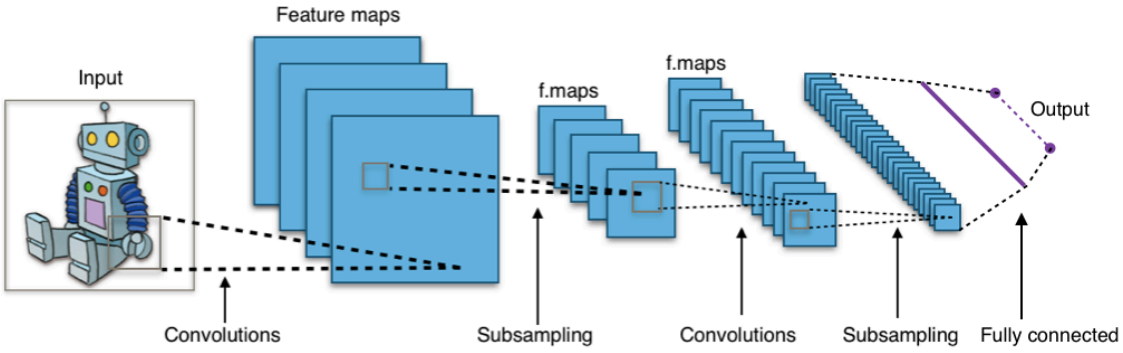


Figure 2.7: A Typical CNN Architecture.

2.7. Activation Functions

An activation function is the non-linearity which determines the output of a neuron in terms of the local field. A number of activation functions are available to be used depending on the type of problems being solved.

2.7.1. Simple Threshold Function

A threshold function is usually a piecewise function which assigns a specific value (usually 1) if the input is greater than or equal to 0, and it assigns zero if the value of the input is less than zero [78]. Mathematically, it can be represented as shown below (see Equation 2.2).

$$f(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases} \tag{2.2}$$

2.7.2. Sigmoid Function

A Sigmoid function is the "S" shape function and is the most commonly used activation function in neural-network constructions. Mathematically, a sigmoid function can be represented as shown in Equation 2.3.

$$f(x) = \frac{1}{(1 + e^{x})} \tag{2.3}$$

Two main limitations of a sigmoid function are that it is not zero-centred and that it offers saturation in the most positive and negative region. Furthermore, because it involves the exponential term, it requires a high computational power.

2.7.3. Hyperbolic Function

A hyperbolic activation function is similar to a sigmoid function, and it addresses one of the limitations of a sigmoid function, which is the inability to be zero-centred. However, due to the saturation, there are chances of getting dead neurons. Mathematically, hyperbolic activation function is represented as shown in Equation 2.4.

$$f(x) = \tanh(x) \tag{2.4}$$

2.7.4. Rectified Linear Unit (ReLU)

A ReLU is the most commonly used activation function with deep nets given that it offers the solution to the problem of computational resources. Mathematically ReLU is expressed as shown in Equation 2.5.

$$f(x) = \max(0, x) \tag{2.5}$$

Because it computes the maximum of the input, it is very fast and efficient. Furthermore, it solves the saturation problem in the positive domain even though the problem remains the same in the negative domain [79]. Networks with ReLU learns many times faster than those that involves saturating nonlinearities. Besides, the desired learning error rate are achieved several times faster in networks with ReLU.

2.8. Loss Functions

A loss function in the CNN are used to measure the performance by comparing the actual and predicted values. In general, a loss function guides the training process. Commonly used loss functions for classification problem include Mean Squared Error (MSE) and Cross Entropy Loss.

2.8.1. Mean Squared Error (L2 Loss)

In the conventional machine learning domain, the MSE is the most commonly used loss function, and it is considered efficient for beginner machine-learning problems [80]. Equation 2.6 presents the mathematical representation of the MSE.

$$\text{MSE Loss} = \frac{1}{n}\sum_{t=1}^{n} e_t^2 \tag{2.6}$$

Where n denotes the number of output classes, and e denotes the difference between the predicted output and the ground truth. In the Euclidean space, the MSE represents a straight line between two points.

2.8.2. Hinge Loss

The hinge loss is another loss function used with the SVM classifier in the machine learning training process. The hinge loss is used for maximum margin classification [81]. Mathematically, the hinge loss can be expressed as shown in Equation 2.7.

$$L(y) = \sum_{i} \max(0, 1 - t_i \cdot y_i) \tag{2.7}$$

Where y_i is the predicted output score of the classifier and t_i is the intended output score of the classifier.

2.8.3. Cross-Entropy Loss

Cross-entropy loss is another commonly used loss function for the classification problems in machine learning, and it performs better than MSE. Cross-entropy loss is based on the maximum likelihood estimation in the statistics [82]. Equation 2.8 presents the mathematical expression for the computation of cross-entropy loss.

$$H_{y'}(y) = -\sum_{t} y_i' \log(y_i) \tag{2.8}$$

Where y_i' denotes the ground truth label of the ith training instance and y_i denotes the predicted label of the ith training instance. Cross-entropy loss is the most commonly used loss function in deep learning. However, its application is different compared to other loss functions.

2.9. Optimisation Methods

2.9.1. Backpropagation

Backpropagation in machine learning is the algorithm of updating the weights of neural networks by propagating the error or loss as feedback. Usually, backpropagation involves the calculation of gradient descent which aims to minimize the loss function and update the weight accordingly. Backpropagation is also referred to as the delta rule for the perceptron for multiplayer feed forward neural networks [83]. Talking about the overall process of training the models and the role of backpropagation, the usual flow of training involves multiple iterations over the input data. For each iteration, the loss function is calculated and the error is back-propagated to update the weights and improve its performance. Hence, the learning or training process involves the loss function to generate errors which are then minimized using the optimization functions and updated based on those optimization network weights or parameters.

Usually, backpropagation achieves this goal by applying the chain rule recursively over the layers and by calculating gradient descent loss with respect to the parameters. This is most commonly optimization approach in backpropagation, an approach known as gradient descent. This approach is discussed in detail in the following sections. Optimized weights of the network are considered as the solution of the learning process. The term “back” in the name refers to the fact that gradient descent is calculated in reverses flow (from output towards input) i.e. the gradient of the final layer is calculated before the gradient of the first layer is calculated. Given the number of computations, training deep networks involves significant computations. However, with the availability of advanced hardware such as GPU, backpropagation computations are no longer a difficult task.

2.9.2. Gradient Descent

Gradient descent is one of the conventional and basic optimization functions used in the backpropagation process, and it involves the optimization of network weights to achieve the local minima of loss functions. This is achieved by moving gradually towards the negative gradient direction of the loss function to achieve the local minima [84]. The aim

of gradient descent is to find the weights (w_k) and biases (b_l) which minimize the cost function given in Equation 2.9.

$$\text{Cost Function} = J\left(\boldsymbol{\theta}: \mathbf{x}^{(i)}, \mathbf{y}^{(i)}\right) \tag{2.9}$$

where $\boldsymbol{\theta}$ denotes the optimized parameter which is to be optimized, $\mathbf{x}^{(i)}$ denotes the input vector for the ith training sample and $\mathbf{y}^{(i)}$ represents the class label for the ith training sample. To minimize this cost function, gradient vectors (∇J) with respect to w_k $\left(\frac{\partial J}{\partial w_k}\right)$ and b_l $\left(\frac{\partial J}{\partial b_l}\right)$ are computed, respectively. Based on the gradient calculations, Equation 2.10 and Equation 2.11 present the update rules for weights and biases, respectively.

$$w_k \rightarrow w_k' = w_k - \eta \frac{\partial J}{\partial w_k} \tag{2.10}$$

$$b_l \rightarrow b_l' = b_l - \eta \frac{\partial J}{\partial b_l} \tag{2.11}$$

Where η denotes the learning rate. By applying above-mentioned update rules repeatedly, all parameters in the trainable networks are optimized to achieve the convergence to a local minima. In practice, gradients for each input sample are computed separately and then averaged as shown in Equation 2.12.

$$\nabla J = \frac{1}{n} \sum_{\mathbf{x}} \nabla J_{\mathbf{x}} \tag{2.12}$$

From the expression, it can be deduced that the calculation of gradients for deep networks involving large numbers of parameters will require a huge computation power and hence will cause the learning process to slow down.

2.9.3. Stochastic Gradient Descent

SGD is basically the extension of the conventional gradient descent algorithm to improve the speed of learning by estimating the gradients ∇J and by computing $\nabla J_{\mathbf{x}}$ for randomly selected small training input samples. This idea of taking the average over small sample leads to better ∇J estimation and helps in speeding the gradient descent computation process

and ultimately the overall learning process [81]. Mathematically, SGD can be expressed as given in Equation 2.13.

$$\frac{\sum_{j}^{m} \nabla J_{\mathbf{x}_j}}{m} \approx \frac{\sum_{\mathbf{x}} \nabla J_{\mathbf{x}}}{n} = \nabla J \tag{2.13}$$

Where m denotes the number of randomly selected input samples, also referred to as mini-batch, and n denotes the total training dataset. The expression in Equation 2.13 can be re-written as follows (see Equation 2.14):

$$\nabla J_{\mathbf{x}} = \frac{1}{m} \sum_{j=1} m \nabla J_{\mathbf{x}_j} \tag{2.14}$$

These calculated gradients from the randomly selected mini-batch are then used to update the weights w_k and biases b_l using the updated rules given in Equation 2.15 and Equation 2.16, respectively.

$$w_k \rightarrow w_k' = w_k - \frac{\eta}{m} \sum_{j} \frac{\partial J_{\mathbf{x}_j}}{\partial w_k} \tag{2.15}$$

$$b_l \rightarrow b_l' = b_l - \frac{\eta}{m} \sum_{j} \frac{\partial J_{\mathbf{x}_j}}{\partial b_l} \tag{2.16}$$

The total of the above equations are computed over the training examples $\mathbf{x}_j$ within the mini-batch. Once one batch is computed, the next mini-batch is selected for training, and this process continues until the whole training data is covered. An epoch is often referred to as a whole iteration of training using randomly selected mini-batches covering the entire training data. Models are trained over multiple epochs to minimize the loss function towards local minima. In the implementation of SGD, three hyper-parameters which significantly influence the training process are as follows:

- **Learning Rate:** It is usually a float value, and it determines the speed of training.
- **Momentum:** It is usually a float value and it influences the rate of damped acceleration in the relevant direction.
- **Decay Rate:** It is the decay of the learning rate over each update.

2.9.4. Adam Optimiser

Adaptive Moment Estimation (Adam) [85] is the method in gradient descent which involves the computation of individual adaptive learning parameters by first and second moments of gradient estimations. Learning rates are adapted based on the average of second moments (un-centered variance) of the gradients. Adam computes the exponential moving average of the gradient and squared gradient. Parameters β_1 and β_2 control the decay of these moving averages. Based on these parameters, updates of first and second moments can be expressed mathematically as given in Equation 2.17 and Equation 2.18, respectively.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.17}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{2.18}$$

The parameters m_t and v_t are usually initialized as vectors of zero and biased towards zero when the decay rates are small. Equation 2.19 and Equation 2.20 present the biased centered first and the second moment estimates of gradients.

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.19}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.20}$$

Based on these moments, parameters of the network are updated using the update rule expressed in Equation 2.21.

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon} \tag{2.21}$$

where $\boldsymbol{\theta}$ denotes the network parameters. For a non-convex optimization problem, the main advantages of using the Adam optimizer are as follows:

- Easy to implement and computationally efficient
- Requires low memory
- Suitable for problems involving huge data and large number of parameters
- Appropriate for problems involving sparse or noisy gradients

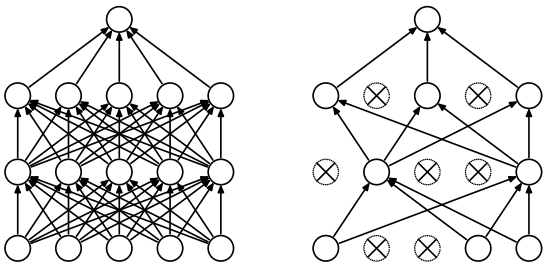
- Hyper-parameters require very less tuning efforts.

2.10. Regularisation Methods

Overfitting occurs when models learn the training data effectively but does not generalise them well enough, thus leading to poor performance over the test data set. In simple words, model learns the training data satisfactory, however, it performs poorly over the unseen testing data. To prevent the problem of overfitting in deep architectures, normalization approaches are used in the literature [86] to generalise the model for unseen data even when the model is trained using smaller datasets and/or with imperfect optimization procedures. This section presents some of the commonly used regularization methods in CNNs.

2.10.1. Dropout

Dropout is one of the most commonly used regularization approach in the deep neural networks to avoid the overfitting problem by leaving or dropping the number of nodes/neuron activations in the network. More specifically, dropout refers to leaving the neuron activations in both visible and hidden layers i.e. temporarily removing the neurons from the network along with all incoming and outgoing connections [87]. Figure 2.8 illustrates the concept of the dropout technique.



(a) Standard Neural Net (b) After applying dropout.

Figure 2.8: Illustration of Dropout Concept (Taken From [87]).

By introducing dropout layers in between, the number of neuron activations are left out which leads to different numbers of architectures, each one of them is trained in parallel and the predictions are averaged. For example, if there are n neurons attached to a dropout layer, there will be 2^n ensemble architectures, and predictions will be averaged over all ensembles. Given that the selection of neuron activations to be left is a random process,

there are no co-adaptations which enable the process to develop meaningful learning feature. The size of computations and model is significantly decreased by introducing dropout layers in the network.

2.10.2. Batch Normalisation

Batch normalization [86] is the approach in which the previous layer activations are subtracted by an averaged batch and divided by the standard deviation of the batch to achieve the stability in the output of neural networks. Batch normalisation adds two trainable parameters in the output activations of layer: standard deviation parameter and mean parameter. Rather than updating all the weights after each iteration, in this approach, SGD performs the de-normalization by changing only two (batch normalization) added parameters for every output activation. Training the model with the pre-trained weights by normalization improves the performance of the pertained model [86]. Batch normalization layers can be added after the dense layer or convolution layer; however, in [86], it was added after the last linear layer.

2.10.3. L1 and L2 Normalisation

L1 and L2 normalisations [88] are the methods involving the parameter regularisation, and the aim to shrink some network parameters by varying regression coefficient to zero to maximise likelihood estimates. Overfitting due to high predictions correlation, which may result in false positive results, is prevented by shrinking the number of parameters. L1 regularization approach helps to achieve feature selection in the sparse feature space by providing an optimal solution when the data are biased and the noise ratio in the data is high. Whenever two predictors are used with high correlation between them, L1 regularization selects one of the two predictors. One significant limitation of L1 regularization is that using it results in the loss of predictive power. On the other hand, L2 regularization leads to small non-zero regression coefficients distributed through the vector space. If two correlated predictors are to be differentiated, L2 regularization keeps both and jointly shrink the coefficients to a small extent.

Regularization methods usually involve an additional term in the loss function. If the loss function aims to minimize the Negative Log Likelihood (NLL), it will be expressed mathematically as shown below (see Equation 2.22).

$$NLL\left(\boldsymbol{\theta}; \mathbf{x}^{(i)}; \mathbf{y}^{(i)}\right) = -\sum_{i=1}^{N} \log P\left(Y = \mathbf{y}^{(i)} \middle| \mathbf{x}^{(i)}, \boldsymbol{\theta}\right) \tag{2.22}$$

where $\boldsymbol{\theta}$ denotes the parameter for which the loss function is computed, $\mathbf{x}^{(i)}$ denotes the input vector for the ith training sample and $\mathbf{y}^{(i)}$ represents the class label for the ith training sample. Using the above relation, the regularization loss can be expressed as shown in Equation 2.23.

$$E\left(\boldsymbol{\theta}; \mathbf{x}^{(i)}; \mathbf{y}^{(i)}\right) = NLL\left(\boldsymbol{\theta}; \mathbf{x}^{(i)}; \mathbf{y}^{(i)}\right) + \lambda R(\boldsymbol{\theta}) \tag{2.23}$$

where $R(\boldsymbol{\theta})$ is the regularization parameter and λ is the hyper-parameter which controls the regularization parameters. $R(\boldsymbol{\theta})$ can be expressed mathematically as follows:

$$R(\boldsymbol{\theta}) = \left|\left|\boldsymbol{\theta}\right|\right|_p^p$$

where

$$\left|\left|\boldsymbol{\theta}\right|\right|_p = \left(\sum_{j=0}^{|\boldsymbol{\theta}|} \left|\boldsymbol{\theta}_j\right|^p\right)^{\frac{1}{p}}$$

Commonly, in neural network, $p = 1$ (L1 norm) is used to shrink the sum of the absolute value of weight. For deep neural networks, $p = 2$ (L2 norm) is used in which the sum of squared of the weights is shrinked. Therefore, regularization is also known as weight decay in the literature.

2.10.4. Early Stopping

Early stopping is an algorithm that is used to determine the best time required by a neural network required to be trained. Early stopping has been proved helpful in preventing the overfitting of huge and complex models [89]. Technically, early stopping is used to monitor the performance of trained models over the validation dataset. Using the early stopping

concept, model training is stopped when the error of model performance increases or shows no further improvement as both are the indicators of overfitting. Early stopping provides the last best copy of parameters (least validation error) which helps in obtaining a better test error for the model [89].

2.11. Classification Method

2.11.1. Softmax Classifier

Inside the CNN, the image has several linear and non-linear operations followed by pooling layers and normalization layers. At the end of the process, we have a feature vector containing raw real numbers that are hardly understandable. The Softmax layer is used to map the feature vector, obtained on the forward pass through the network, into a vector containing the probabilities of $\mathbf{x}$ to belong to each class. The Softmax layer is a generalization of the logistic function, and it performs a mapping $\mathbb{R}^d \rightarrow \mathbb{R}^d$ where each element of the output is within the range $(0, 1)$ and $\sum_i^N \text{softmax}(\mathbf{x}_i) = 1$. Its formula is given by $\text{softmax}(\mathbf{x}_i) = \frac{e^{\mathbf{x}_i}}{\sum_j^N e^{\mathbf{x}_j}}$

This layer is the main part of multi-label classification problems as it allows us to compare the distribution of probabilities returned by the CNN and the ground-truth labels.

The last layer of a CNN for image classification tasks is the Softmax layers and is used to normalize a real value vector of K dimensions into a vector between 0 and 1. Softmax is the generalization of binary logistic regression classifier for the multiple classes. In terms of loss function, Softmax loss can be expressed as shown in Equation 2.24.

$$L_i = -\mathbf{f}_{y_i} + \log \sum_j e^{\mathbf{f}_j} \tag{2.24}$$

where $\mathbf{f}$ is the vector of class scores which is usually represented as $f(\mathbf{x}_i; \mathbf{W})$ and $\mathbf{f}_j$ represents the jth element of that vector. The output of the Softmax function represents the probabilities of an image belonging to certain class j. Mathematically, the softmax function can be illustrated as shown below (see Equation 2.25).

$$\sigma(\mathbf{z})_j = \frac{e^{\mathbf{z}_j}}{\sum_{k=1}^{K} e^{\mathbf{z}_k}} \tag{2.25}$$

where $\mathbf{z}$ is a k dimensional vector of real values and $\sigma(\mathbf{z})$ is the output probability vector. Because Softmax guarantees a well-behaved probability distribution function without losing generality, we can use the σ distribution provided by the output of the Softmax layer as an argument for this expression from above to calculate the cross-entropy loss function as shown in Equation 2.26.

$$L_i = \frac{1}{N}\sum_{i=1}^{N} H(p_i, q_i) = -\frac{1}{N}\sum_{i=1}^{N} p(\mathbf{x}_i)\log\big(q(\mathbf{x}_i)\big) \tag{2.26}$$

Where $p(\mathbf{x}_i)$ is the data, $q(\mathbf{x}_i)$ is the estimated distribution of the variable and N denotes the dimensions of the feature space. Particularly, this calculation will be performed over mini-batch for the feature vectors $\mathbf{x}_i$ with the d dimensions. Equation 2.26 can be expressed as given in Equation 2.27.

$$L_i = -\frac{1}{N}\sum_{i=1}^{N} \log\left(\frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}\right) = -\frac{1}{N}\sum_{i=1}^{N} \log\left(\frac{e^{w_i^T x_i + b_i}}{\sum_{j=1}^{k} e^{w_j^T x_j + b_j}}\right) \tag{2.27}$$

2.11.1.1. Categorical Cross-Entropy Loss Function

In order to measure the performance of the model in multi-label classification problems the categorical cross-entropy loss function is used (also named negative log-likelihood, NLL). Considering $M(\mathbf{x}_i)$ as the output of the model before the Softmax layer, this loss functions is defined as

$$L_{CE} = -\sum_{i}^{N} y_i \log(\text{softmax}(M(\mathbf{x}_i)))$$

where $\mathbf{y}_i$ is the ground-truth vector for the image $\mathbf{x}_i$ in one-hot encoding format, $\log(\text{softmax}(M(\mathbf{x}_i)))$ will give us the log-probabilities of $\mathbf{x}_i$ to belong to each class of the dataset. Since it is multiplied to $\mathbf{y}_i$, it will only account for the log-probability of the ground-truth class. Therefore, if $\text{softmax}(M(\mathbf{x}_i))$ tends to zero, then the error L_{CE} tends to ∞ (because the model assigned a probability of almost zero to the ground-truth class), while

if $\text{softmax}(M(\mathbf{x}_i))$ tends to one, then L_{CE} tends to zero (because it correctly assigned a probability of one to the ground-truth class).

2.11.2. Support Vector Machines as the Classification Layer of CNN

SVM use a different loss function, called hinge loss. This loss function main goal is to find the hyperplane with the biggest support between samples of different class. Using a 2D geometrical interpretation, this means, that the hinge loss will find the line that splits different semantic label while it has the biggest possible distance between samples of different classes. The SVM loss function is

$$\mathcal{L}_{SVM} = -\sum_{j \neq i}^{N} \max\left(0, M\left(\mathbf{x}_j\right) - M(\mathbf{x}_i) + 1\right.$$

which accumulates the differences on the classification between the ground-truth class $M(\mathbf{x}_i)$ and the rest of the classes $M\left(\mathbf{x}_j\right)$. In the case that the model returns the highest value to the ground-truth class $M(\mathbf{x}_i)$ then the loss will have a value between [0, 1]. if $M(\mathbf{x}_i)$ is not the highest probability then the loss will accumulate the differences between the values returned for all the classes against the ground-truth class. Note that the value of $M(\mathbf{x}_i)$ or $M\left(\mathbf{x}_j\right)$ is not necessarily a probability. Usually, SVM use a linear function applied to the feature vectors before computing the hinge loss. However, there are other possible kernel approaches that might give better result. For example, in this project we have used a quadratic radial basis function that maps the input into a higher dimensional space which is supposed to make easier the classification problem. Therefore, applying this function we are capable to get better performance. Moreover, this function has a free parameter that can be trained in order to obtain higher accuracy rates.

Adding non-linearities SVM by themselves do not add any non-linearities to the input features. That means that its separability and, therefore, the performance of the model will depend only on the quality of those features. In some cases, it can be helpful to add a previous step (before classification) that applies a non-linear kernel to the input in order to map it into a new feature space. In this project we have explored two different kernels: the linear kernel and the radial basis function. Considering x and y as the input and the ground-

truth, the linear kernel is defined as $k(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^T \boldsymbol{y} + c$ where c is an optional constant. On the other hand, the radial basis function kernel is defined as a quadratic mapping of the input $k(\boldsymbol{x}_i, \boldsymbol{y}_i) = e^{-\frac{\|\boldsymbol{x}_i - \boldsymbol{y}_i\|^2}{2\sigma^2}}$ where σ is a trainable parameter.

In this project we are going to compare the performance of CNNs using Softmax classification on top against using a SVM to classify the feature vectors extracted from a CNN model that has been fine-tuned using a triplet loss function.

2.12. Deep Learning Approaches for Drowsiness and Distraction Detection

This section presents the details of the existing deep learning approaches based on computer vision and machine learning for the distraction and drowsiness detection among drivers. Deep learning approaches involve the extraction of features automatically from the entire input image to facilitate the detection. This is often achieved through the use of machine learning approaches.

Dwivedi et al. [90] proposed a drowsiness detection algorithm based on the representation learning technique. In machine learning, representation learning techniques have been proposed and used to automatically discover (through learning) the key discriminative features (representation) from raw data to perform detection and/or classification tasks. CNN-based deep learning architecture was used to learn and extract drowsiness related facial features. Both latent and complex non-linear facial features which include local receptive fields, sharing of weights and sometimes spatial or temporal pooling were extracted by the deep architecture; all weights are learnt, all the learned weights acts a learned feature detectors for driver drowsiness and these feature detectors are convolved with input images to produce the final features used for classification in the last layer, Softmax was used to classify the features in order to decide the drowsiness state of the driver. The proposed algorithm was trained and evaluated over the customized dataset since the standard dataset in this regard was not available. Data was collected from 30 different subjects, including the diversity of skin tones, eye colors, and eye shapes. Authors achieved

the classification accuracy of 92% and 78% for the validation data and real-driving data, respectively.

Park et al. [91] proposed a driver drowsiness detection system that utilised multiple deep neural networks. The proposed deep learning architecture was named as Deep Drowsiness Detection (DDD). A camera sensor was used to capture RGB videos of the driver. Three different deep neural networks were used to learn the facial features, head pose and background illumination variations. The outputs from of all three networks was concatenated and categorised by a Softmax classifier. Authors evaluated the proposed algorithm on the NTHU-drowsy driver detection benchmark dataset which consists of 640×480 videos collected under Infrared (IR) illumination. 22 different subjects of both genders and different ethnicities have been recorded with various facial characteristics. The database consists of training and evaluation sets: training set contains 360 video clips of 18 subjects, while the evaluation set consists of 20 video clips of 4 subjects and achieved the overall detection accuracy of 73.06%.

Zhang [92] suggested and compared two different machine learning classifiers, SVM and CNN. In the SVM based distraction detection algorithm, the PCA approach was used to reduce the data dimensionality and produce inputs to the SVM. For the CNN based algorithm, the VGG architecture was trained using the transfer learning approach rather than the scratch approach. Both algorithms were evaluated with the Kaggle distracted driving dataset; after that, the results were compared The Kaggle scores of 1.53 and 0.22 were reported for SVM-based and CNN-based algorithms, respectively. Although CNN-based algorithm proved more efficient in distraction detection; however, the problem of overfitting was observed.

Mbuvha and Wang [93] also recommended CNN-based distraction detection algorithm in solving the Kaggle challenge of distracted driving. The transfer learning of two deep architectures, VGG and AlexNet, was performed. Both deep architectures were modified according to the requirements of Kaggle challenge and were evaluated based on the dataset provided by Kaggle. Authors achieved an accuracy of 98.2% and 99.7% for off-the-shelf classifier and fine-tuned CNN classifier, respectively. Furthermore, authors evaluated the

performance of classifiers with the split data set where training and testing data was divided among different drivers. A significant degradation from 99.7% to 55.9% was observed for the fine-tuned CNN algorithm. As a result, the authors concluded that the CNN algorithm is not efficient for driver invariances, and overfitting may be observed if the algorithm is subjected to more training.

Venturelli et al. [94] proposed the deep CNN-based head pose estimation algorithm to estimate the yaw, pitch and roll angles of head. The group of researchers used the Microsoft Kinect camera sensor to accurately capture the depth maps of drivers. Rather than extracting facial features, they solved the problem as a regressing problem, which involves the estimation of mapping function to map given input variables into continuous output variables and extracted the head position (head angles) directly from the depth images. Authors used Stochastic Gradient Descent (SGD) optimization algorithm to resolve the back-propagation and L2 loss function. The proposed algorithm was evaluated with the Biwi Kinect Head Pose dataset, and promising results were achieved. The group of scholars achieved estimation angles of 2.8 ± 3.1, 2.3 ± 2.9 and 3.6 ± 4.1 for head pitch, head roll and head yaw angles respectively. Furthermore, a processing time of 10 milliseconds per frame was achieved by the proposed algorithm.

Streiffer et al. [95] developed a deep architecture known as DarNet to detect the distracted behaviours of drivers autonomously. DarNet is a unified data collection and analysis platform which can automatically detect and classify the distracted behaviours. The proposed DarNet architecture consisted of two modules: a data collection system and an analysis engine. An in-vehicle mounted camera sensor was used to collect the frontal facial information of the driver. The analysis engine comprises a CNN-based deep architecture, which can classify the distraction state. The proposed system was evaluated with the datasets collected from five different drivers. From the experiments, Top-1 classification accuracies for normal and down-sampled datasets are 87.02 and 80%, respectively.

Dellinger et al. [96] proposed a computer vision-based algorithm for the autonomous detection of secondary driving tasks. They captured the visual information of face, steering wheel and speed pedal to detect if a driver was involved in secondary driving tasks e.g.

talking on the phone or texting on the phone. Two different algorithms were proposed in this research: the Histograms Oriented Gradient (HOG) with the SVM algorithm and the deep learning based CNN architecture. HOG with the SVM algorithm was used to detect the presence of passengers and foot over pedal using the information from the cockpit camera and pedal camera. A RI was found from the input data sequences and then HOG descriptors were computed. At the next stage, the computed HOG descriptors were classified by a pre-trained SVM classifier to decide if the passenger was present and if the foot of drivers was over the pedal. Deep learning-based algorithm was used to detect if the driver was texting or putting the phone close to the ear and if the driver's hands were on the steering wheel. Algorithms were evaluated over a large dataset of 48 videos captured by authors under diversity of conditions such as day light condition, night driving condition, sunny day condition, rainy day condition and different drivers. From the experimental results, the detection accuracy of 95.6% and about 99% was achieved for the presence of passengers and foot over pedal using HOG with SVM. Furthermore, the detection accuracies of 99.5%, 21% and 18% were achieved by the deep learning approach for hands on steering wheel, texting and phone to the ear detections, respectively. Lower detection accuracies for texting and phone to the ear detections were because of unbalanced dataset.

Masala and Grosso [97] presented a real-time driver-attention monitoring system using machine learning robust classifiers. A combination of a binary classifier and neural network based data reduction was used to detect the attention level among drivers. The overall system consisted of two stages. At the first stage, the head pose is detected for each frame. At the second stage, eye blinking is detected. Viola and Jones algorithm was used to detect the face and eyes from the input image sequences. Feed Forward Back Propagation (FF-Bp) neural network was used to classify the attentive behaviours from the inattentive behaviours, based on the extracted information of head pose and eye blinking. The proposed head pose detection and eye detection algorithm were evaluated over the IDIAP Head Pose Database and the FERET dataset, respectively. An accuracy of 92% and 81% was achieved for pose detection and eye detection, respectively.

Abouelnaga et al. [98] developed a public dataset for distracted driver posture estimation. Furthermore, they proposed a deep learning-based distraction classification system to

detect distracted driving behaviours efficiently. Genetically-weighted ensemble of classifiers were used in the proposed system. Two deep architectures, AlexNet (from scratch) and Inception V3 (transfer learned from ImageNet), were trained over the dataset. Distraction related facial features such as face orientation, hand positions and skin segmentation were classified by the deep networks. Proposed algorithms were evaluated over the established dataset, which was divided into two groups: 75% for training and 25% for testing. classification accuracies of 93.65% and 95.17% were achieved with AlexNet and Inception V3 architectures without implementation of genetically weighted at classification layer, respectively. Furthermore, the genetically weighted ensemble of CNN achieved classification accuracy of 95.98%.

Hssayeni et al. [99] suggested a distraction detection system for drivers using computer vision and machine learning approaches. An in-vehicle dashboard camera was used to capture the visual information of the driver. The aim was to classify the distracted behaviours into one of seven classes (one of safe and six of distracted driving). Two techniques, handcrafted features with SVM classification and deep CNN, were implemented and compared. For handcrafted features, a blend of HOG and Scale Invariant Feature Transform (SIFT) descriptors were used. On the other hand, for deep learning architecture, transfer learning of AlexNet, VGG-16 and ResNet-152 were performed for the distraction detection tasks. To evaluate the proposed algorithms, an online dataset of over 20,000 images [add reference] was used, including the images of safe and distracted driving. The overall dataset was divided into two parts: 80% images were used for training and the remaining 20% were used for testing. From the experimental results, very low accuracies of 33.2% and 21.5% were achieved by HOG and SIFT handcrafted features with SVM classifiers. For deep architectures, classification accuracies of 72.6%, 82.5% and 85% were achieved by AlexNet, VGG-16 and ResNet-152, respectively.

Table 2.6 presents a comparison of the above cited literature regarding distraction and drowsiness detection among drivers using deep learning approaches. Furthermore, Table 2.7 presents the categorization of cited literature at much higher level.

Critical analysis of comprehensive literature on drowsiness and distraction detection in drivers revealed that use of deep learning approaches has improved the overall detection accuracies and generalization problem. Conventional approaches were able to detect drowsiness and distraction based on the local hand crafted facial features, however, would not be able to detect distracted behaviours such as use of mobile phone. Where deep learning approaches provided the solution for the challenge. As a result, various deep networks have been implemented for the detection of distraction detection in drivers as a part of research presented in this thesis. Chapter 5 and Chapter 6 present the theoretical and implementation details of deep learning based distraction detection in drivers.

Table 2.6: Comparison of Literature Related to Drowsiness and Distraction Detection using Deep Learning Approaches.

Authors	Database Information	Features and Measures	Feature Extraction Method	Classifiers	Performance
Dwivedi et al [90]	Custom collected data from 30 subjects	Facial Features	Face	SoftMax CNN	For Training Dataset: 92% For Random Dataset: 78%
Park et al. [91]	NTHU-drowsy driver detection benchmark dataset	Facial Features, and Head pose	Face, eyes and head	Softmax CNN	73.06%
Zhang [92]	Kaggle Distraction Challenge Dataset	Distraction related features	Deep Learning Approach	SVM and VGG CNN Architecture	**Kaggle Score** SVM: 1.53 VGG CNN: 0.22
Mbuvha and Wang [93]	Kaggle Distraction Challenge Dataset	Distraction related features	Deep Learning Approach	VGG and AlexNet	Off-the-shelf VGG: 98.2% Fine-tuned AlexNet: 99.7%
Venturelli et al [94]	Biwi Kinect Head Pose Dataset	Head Pose	Neural Networks	Deep CNN	Pitch angle: 2.8 ± 3.1 Roll angle: 2.3 ± 2.9 Yaw angle: 3.6 ± 4.1
Streiffer et al. [95]	Dataset Collected from 5 different subjects	Frontal Facial Features and IMU sensor data	Deep Learning Approach	DarNet deep architecture	Normal data: 87.02% Down-sampled data: 80%
Hssayeni et al. [99]	Online dataset of over 20,000 images	Distraction related facial features	HOG and SIFT	SVM, AlexNet, VGG-16 and ResNet-152	HOG+SVM: 33.2% SIFT+SVM: 21.5% AlexNet: 72.6% VGG-16: 82.5% ResNet-152: 85%
Dellinger et al. [96]	Custom collected dataset	Facial information, steering wheel and speed pedal	HOG and Deep Learning	SVM and Deep CNN	**HOG+SVM** Passenger: 95.6% Foot over pedal: 99% **Deep CNN** Hand on wheel: 99.5% Texting: 21% Phone with ear: 18%
Masala and Grosso [97]	IDIAP Head Pose Database and FERET	Head pose and eye blinking	Viola and Jones	Feed Forward Back Propagation Neural Network	Pose detection: 92% Eye detection: 81%
Abouelnaga et al. [98]	Proposed Custom Dataset	Face orientation, hand position and skin segmentation	Deep Learning Approach	AlexNet, Inception V3 and Genetically weighted CNN	Alexnet: 93.65% Inception V3: 95.17% Genetically weighted CNN: 95.98%

Table 2.7: Categorization of Deep Learning Based Approaches from Literature for Distraction and Drowsiness Detection.

	CNN Model					**Extracted Features**			**Extraction Approach**		**Classifier**	
	AlexNet	VGG	ResNet	Inception	others	Facial	Head and body Pose	In-Vehcile	Handcrafted	CNN	Softmax	SVM
Drowsiness	[91]	[91]	–	–	[90] [91]	[90] [91]	[91]	–	[90]	[91]	[90] [91]	–
Distraction	[93] [98] [99]	[92] [93] [99]	[99]	[98]	[94] [95] [96] [97]	[95] [96] [99]	[92] [93] [94] [97] [98] [99]	[95] [96]	[97] [99] [92]	[92] [93] [94] [96] [98] [99]	[92] [93] [94] [98] [99]	[99] [92]

CHAPTER 3

CONVENTIONAL APPROACHES

3.1. Introduction

Drowsiness detection using computer vision approaches often involves the combination of different detection algorithms for different visual features within multiple local regions of the human body such as the face, eyes, mouth, and head pose detections. Features related to drowsiness and distraction from each detection algorithm are extracted using mathematical measures such as PERCLOS, Degree of Openness (DOO) [100] and HWrate [101] and can then be categorized using a SVM classifier. These approaches are referred to as conventional vision approaches as their popularity has been overtaken by the deep learning approaches mainly due to their better performance. The conventional approaches extract and rely on hand-crafted features whereas the deep learning approaches learns from a huge dataset the best features for a given task. This chapter outlines the theoretical and mathematical basis of conventional vision approaches used for detecting drowsiness and distraction among drivers. Furthermore, this chapter provides the details of SVM classifications using conventional vision approaches in the context of drowsiness and distraction detection. The first year of my MPhil research, mainly focused on conventional vision approaches and methodologies used in detecting drowsiness and distraction among drivers. This part of my early research will be discussed more elaborately in this chapter.

3.2. Face Detection

Face detection is one of the most significant elements in detecting drowsiness among drivers using the visual information because most drowsiness-related features which are required for extraction are related to the facial region. Hence, it is of paramount importance that at the first stage, the facial region is accurately identified and isolated from the input images.

Evidence from the literature indicates that the following are the most commonly used face-detection approaches.

- **Knowledge Base Face Detection:** This approach uses previous knowledge of the characteristics and locations of different facial features such as eyes, nose and mouth. The relationship of these facial features is used to identify a facial region.
- **Template Matching Base Face Detection:** This method uses the pre-stored images of different types of face in the database and computes the similarity of captured images by finding the correlation between the captured images and stored images to decide whether the captured image is a face or not.
- **Appearance Base Face Detection:** This technique utilises machine learning and statistical models trained over large datasets to classify the captured images as facial or non-facial images. These algorithms usually use approaches such as SVM, CNN, and deep learning.

In this research, face detection was achieved using the knowledge-based Viola and Jones face detection algorithm. The following section presents theoretical details regarding the Viola and Jones face algorithm.

3.2.1. Viola and Jones Face Detection Algorithm

Viola and Jones algorithm [102] is commonly used in literature for efficient facial-region detection in real-time situations. There are mainly three steps or elements involved in the Viola and Jones object detection algorithm, and they are as follows:

- Integral Image Representation
- Haar-Like Feature Computation
- Classification

In the first step, the input image is depicted as an integral representation, and then Haar-Like features are computed from the integral image representation. Generally, Haar-Like features represent the difference of intensity values between different regions of the face. Figure 3.1 shows three 2D-Haar-like features for facial features proposed by Viola and Jones for face detection.

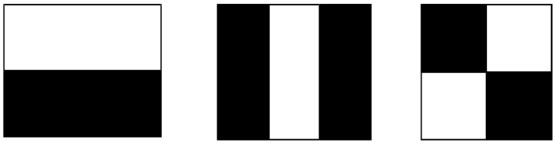


Figure 3.1: Haar-Like Features Proposed by Viola and Jones for Face Detection.

The simple idea of taking difference of the sum of pixels in the white region and that of pixels in the black region is used to compute the Haar-Like features. For example, if X_i denotes the black region and Y_j represents the white region, then the feature vector V_h can be computed by using the mathematical expression presented in Equation 3.1.

$$V_h = \sum_{i=1}^{N_a} X_i - \sum_{j=1}^{N_b} Y_j \tag{3.1}$$

where N_a denotes the total number of pixels in the black region, while N_b represents the total number of pixels in the white region. The integral image at any point is basically the sum of pixels above and to the left of that point. For example, the integral image at point (x_1, y_1) will be denoted as $X_{sum}(x_1, y_1)$ and can be computed using the mathematical expression presented in Equation 3.2.

$$X_{sum}(x_1, y_1) = \sum_{x_1=1}^{N_1'} \sum_{y_1=1}^{N_2'} X_{orignal}(x_1, y_1) \tag{3.2}$$

where $X_{orignal}(x_1, y_1)$ denotes the pixel value of the original input image.

AdaBoost using haar classifier is the single rectangle feature and threshold that best separates positive (faces) and negative (non-faces) training examples, in terms of weighted error, its used as an input feature for cascade classifier.

Therefore, in order to select them effectively, it is prudent to weigh, rank and train the weak classifiers. Rather than a single classifier, a group of AdaBoost-based classifiers is used in the Viola and Jones algorithm. This group of AdaBoost-based classifiers is referred to as weak classifiers. Figure 3.2 illustrates the structure and the working principle of cascade

classifiers. At the final stage, the features selected through the weak classifiers (reduced significantly in numbers) are selected/classified by a strong classifier.

At final stage of Viola and Jones algorithm [102], a strong cascade classifier has been used to select some of the features which are not rejected by initial weak classifiers. Figure 3.2 presents the generalized structure of the cascade classifier. Equations 3.3 and 3.4 shows the mathematical representation of the strong classifier.

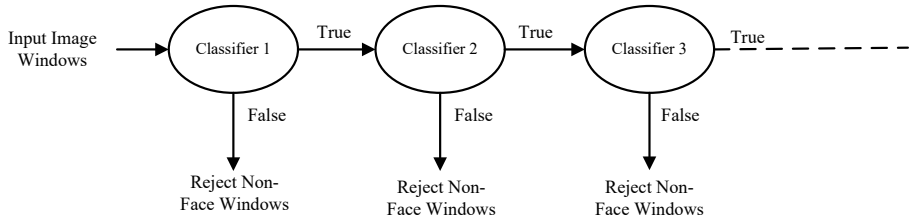


Figure 3.2: Cascade Classifier Structure of the Viola and Jones Algorithm.

$$h(\mathbf{x}) = \text{sgn}\left(\sum_{j=1}^{M} \alpha_j \, h_j(\mathbf{x})\right) \tag{3.3}$$

$$\alpha_j = \frac{1}{2}\log\left(\frac{1-\epsilon_j}{\epsilon_j}\right) \tag{3.4}$$

were $h_j(\mathbf{x})$ represents the weak classifiers, M represents the total number of classifiers in the cascade structure and ϵ_j denotes the weighted error rate of the weak classifier. The use of weak classifiers at the initial stage of the algorithm helps in rejecting sub-windows in the image that do not contain a face. The strong classifier at the end of algorithm assists in selection regions with the possibility of a face [102, 103]. Figure 3.3 presents the block diagram of different steps involved in the Viola and Jones face detector algorithm.

Figure 3.3: Block Diagram of Steps Involved in the Viola and Jones Object Detection Algorithm [102].

The Viola and Jones object detection algorithm is considered one of the most efficient and effective algorithms for detecting objects in real-time and is often used by the researchers in the field of computer vision for this purpose. The idea of using the integral image representation significantly improves the feature computation process. Furthermore, from a large number of computed features, cascade classifiers can reject a significant number of negative sub-windows or sub-windows in the image that do not contain a face, thus making the final classification process faster. The purpose of cascade classifiers is to focus only on regions in an image where the possibility of a face is maximum. However, real-time face detection is a relatively complex task, and its performance significantly depends on the number of factors such as illumination conditions, face pose variations, the person's skin, the presence of the eyeglasses and occlusion.

3.3. Eye Detection

The eyes are considered one of the most significant facial features in detecting drowsy driving. Also, eye blinking and gaze direction are prominent measures used in ascertaining whether a driver is susceptive to drowsy driving. Based on the illumination, two approaches are used in the literature for eyes detection. One is the IR-based detection, and the other is natural light-based detection. In the IR approach, the eyes are exposed to near-infrared light and synchronous approximation, and pupil reflection properties are studied to effectively track the eyes. Furthermore, IR lighting conditions help in estimating the eye gaze [104]. In the natural light-based eye detection method, only active natural light is used to track the eyes. However, eye detection in natural light is relatively complex because of multiple reflections and shadows.

In this literature, localization of the eyes on the detected facial region is achieved by using the face anthropometric properties derived from a face database analysis [105]. From the facial RI, two rectangles containing the eyes are obtained. RI_L is used to denote the left eye rectangle and RI_R the right eye rectangle. Figure 3.4 presents the visual illustration of the implemented concept for eye detection using Viola and Jones.

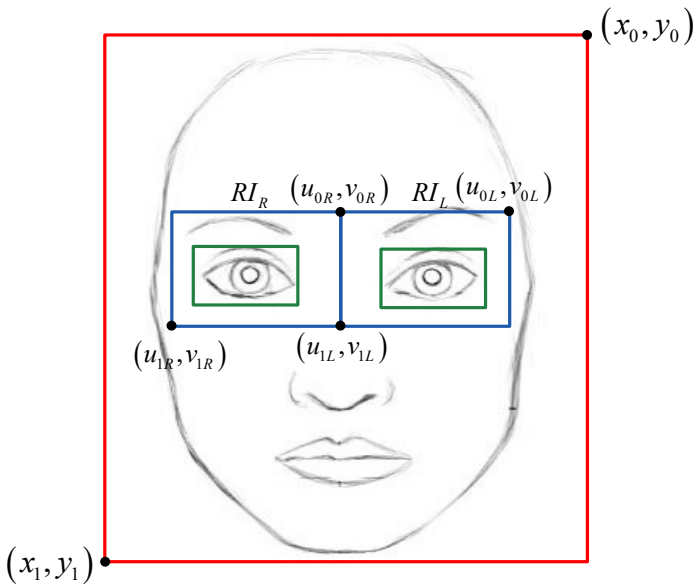


Figure 3.4: Demonstration of the eye Detection with Region Parameters.

Once the eyes region are identified as rectangles, the exact eye location is computed using the set of equations presented in Equation 3.5 [106] and isolated by small rectangles within large rectangles. (x_0, y_0) is taken as the top left point of the large rectangle, while (x_1, y_1) is taken as the right bottom point of large rectangle.

$$(u_{0L}, v_{0L}) = \left(x_0 + \frac{w}{6}, y_0 + \frac{h}{4}\right)$$

$$(u_{1L}, v_{1L}) = \left(x_0 + \frac{w}{2}, y_0 + \frac{h}{2}\right)$$

$$(u_{0R}, v_{0R}) = \left(x_0 + \frac{w}{2}, y_0 + \frac{h}{4}\right)$$

$$(u_{1R}, v_{1R}) = \left(x_1 - \frac{w}{6}, y_1 + \frac{h}{2}\right) \quad (3.5)$$

Where $w = x_1 - x_0$ and $h = y_1 - y_0$ denote the width and height of the respective region of the rectangle for eye detection. (u_{0L}, v_{0L}), (u_{1L}, v_{1L}), (u_{0R}, v_{0R}) and (u_{1R}, v_{1R}) represent the top left and bottom right corner points for the left eye rectangle and right eye rectangle, respectively. Head movements, illuminations changes do not usually allow for complete incorporation of information from grey level pixels, so we use pixel information from a

random sample, grey-level pixel values are incorporated by obtaining a random sample and then adjusting it to the defined parametric model. In this way, the exact eye position is obtained [106].

It has been reported in the symptoms of drowsiness section presented in Section 2.3 of this thesis that the eye-blinking rate of a driver varies differ between active and drowsy conditions. Therefore, eye blinking is used as a criterion for deciding the level of drowsiness among drivers. In the literature, the term used to represent the eye blinking rate is called the PERCLOS [48, 54] and considered one of the most valid ocular parameters for drowsiness detection. Basically, the PERCLOS defines the amount of time needed to completely open or close the eyes (speed of blink), and it can effectively be used as drowsiness monitoring measure. In this research, the idea that the eyes close when the pupil is occluded by the eyelids has been used for the calculation of PERCLOS. Figure 3.5 presents the principle used in this research to determine the PERCLOS.

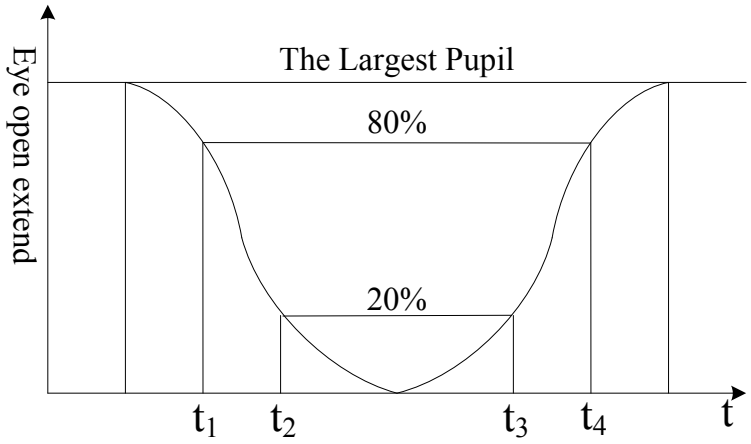


Figure 3.5: Principle of PERCLOS Computation Proposed by the Weijie et al. [107].

In Figure 3.5, time $t_1 - t_2$ denotes the time of the eye closing from 80% to 20%, time interval $t_2 - t_3$ denotes the time duration for which the eyelid remains at 20% opened or closed, while time interval $t_3 - t_4$ denotes the time of the eye opening from 20% to 80%. Based on these time intervals, the PERCLOS is computed using the expression presented in Equation 3.6.

$$\text{PERCLOS} = \frac{t_3 - t_2}{t_4 - t_1} \times 100 \tag{3.6}$$

Furthermore, along with the PERCLOS, this research incorporates the idea of calculating the height and width of the eye for each frame as proposed by the Weijie et al. [107]. Figure 3.6 indicates the visual illustration for the idea of using the eye height and width as a measure for eye openness. A term known as the EyeHWRate (μ) has been used and calculated using the expression as presented in Equation 3.7. If the $\mu < 27\%$, the eye is considered a closed state and if the $\mu \geq 40\%$, the eye is regarded as an open state [107].

$$\mu = \frac{H}{W} \times 100 \tag{3.7}$$

where H denotes the height of the eye and W denotes the width of the eye.

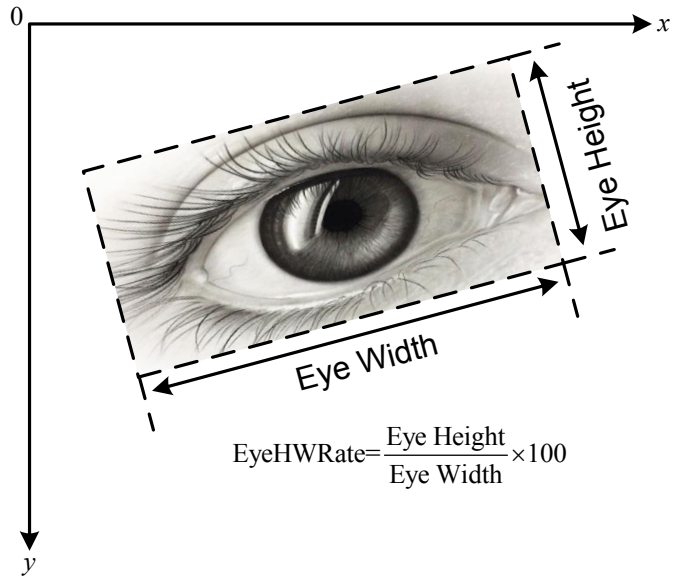


Figure 3.6: The Ratio of the eye-Height and Eye-Width Proposed by Weijie et al. [107].

3.4. Yawning Detection

Yawning detection and the level of mouth openness are considered critical features in estimating the drowsiness level of drivers. Yawning detection at its first stage involves the detection and localization of the mouth. A number of approaches are available in the literature for yawning detection. In a study carried out by Abtahi et al. [108] yawning is

detected in two steps independent of the mouth location in the facial region. At the first stage, a hole is detected within the facial region due to a wide opening of the yawning mouth; at the second stage, the detected hole is verified if it is within the region where the mouth is localized in a facial image. Hariri et al. [109] detected the yawning by utilizing features such the rate of change in mouth contour during yawning and mouth area aspect ratio. In this literature, following are the methods which have been used to detect the mouth and yawning among drivers. Three main steps are involved. The first is the detection of the mouth region. The second is the detection of mouth related features, and the last is the determination of the degree of mouth openness.

Detecting the mouth: Before determining the angle of mouth openness, a system must be able to detect the mouth in the facial images correctly. To estimate the position of the mouth on the face, previous knowledge can be used such as knowing that the mouth is always present in the lower region of the face and being knowledgeable about the distance between the lips corner and the lips from chin [100].

Detecting the features of the mouth: The next step after the detection of the mouth region is to detect the important features of the mouth for which in this literature a projection-based approach is used. Given that the mouth variations are much dominant in the vertical direction, the lips corners are detected using the vertical differences. Applying the thresholding technique will result into two columns of the mouth having lips corner points. These points define the mouth and can be used to determine the orientation of the mouth [100].

Degree of mouth opening: It is defined by the reaction of the driver's mouth during the yawning state and is used to calculate the exact mouth angle. It is represented with DOO_m. Using the mouth model presented in Figure 3.7, DOO_m can be represented mathematically as given in Equation 3.8.

$$DOO_m = \frac{h}{w} = \frac{h' \times \cos\theta}{w} \tag{3.8}$$

where w represents the width of the mouth (the distance between lips corners), h represents the mouth height (the distance between the upper and the lower boundary) and θ represents the orientation angle of the detected mouth. The basic block diagram of the mouth detection and respective results are shown in Figure 3.8.

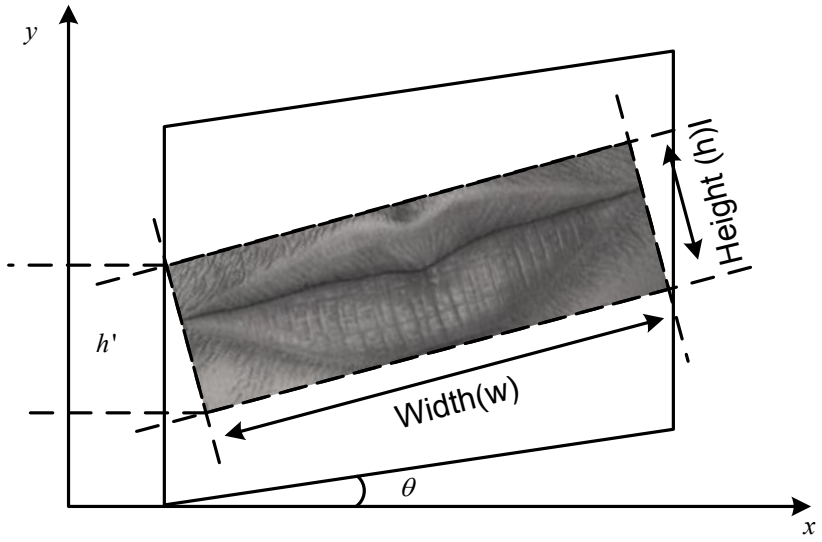


Figure 3.7: Mathematical Model of Mouth Proposed by Wang et al. [100].

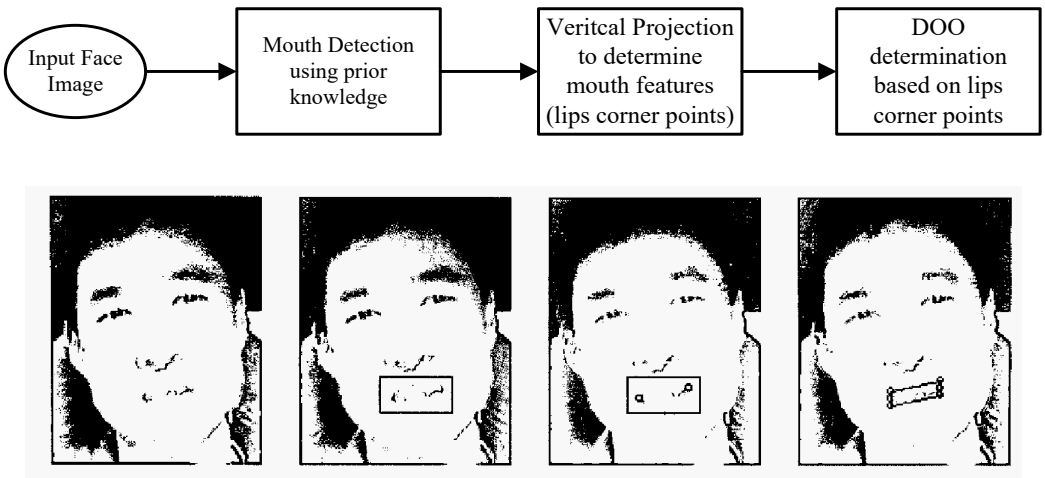


Figure 3.8: General Block Diagram and Respective Output for Mouth Detection [100].

In the next step after detecting the mouth and determining the degree of mouth openness, the model is to estimate the yawning state of the driver by measuring the height of the mouth (mouth opening). It has been reported by Zainal et al. [101] that during yawning, the

height of an individual's mouth is increased by a specific value. Hence, based on the respective threshold value for a normal mouth height, yawning can be determined in terms of yes or no by a simple comparison. Zainal et al [101] note that every individual has different facial features. They also commented that a common threshold cannot be used; rather, by taking the average of 50 frames, an efficient threshold value can be calculated which will result in a more accurate classification of yawning. Expressions for threshold and yawning estimations are presented in Equation 3.9 and Equation 3.10, respectively.

$$\text{Threshold}_{\text{Mouth}} = \text{MouthHeight}_{\text{Avg of 50 frames}} + \frac{\text{MouthHeight}_{\text{Avg of 50 frames}}}{3} \quad (3.9)$$

$$\begin{cases} \text{Yawning}, & \text{current height} > \text{Threshold}_{\text{Mouth}} \\ \text{Not Yawning}, & \text{Otherwise} \end{cases} \quad (3.10)$$

3.5. Head Pose Detection

The head pose is another important feature which is commonly used in drowsiness and distraction systems for determining the attention level of drivers. Mainly, the decision on the alertness is made based on whether the driver is looking straight or not - a factor that is determined by measuring the head tilt angle. If the head tilt angle of drivers is greater than a certain value, the driver is classified as the in-attentive. The gaze direction is an important feature for connecting with the head pose, and it is used to determine the attention level. Head pose detection is integrated usually because it is helpful in situations where the facial features are not visible, such as the driver looking sideways or looking down. In such case, head pose detection is used to alert the driver about its attentiveness. Kang et al. [110] present a comprehensive review of different head pose detection techniques. Furthermore, Chapter 2 includes a complete section of the literature review about head-pose detection techniques.

For the purpose of this research, a simple idea has been implemented based on the position of the head whether it is downward or straight. If it is downward, it is classified as distracted; otherwise, it is regarded as attentive. For this purpose, a trained SVM classifier has been used with the distraction dataset. Also available in the literature there are many

comprehensive approaches, which can effectively be used for head pose detection. A detailed description of the head-pose detection mechanism used in this research is presented in Chapter 4.

3.6. Support Vector Machine (SVM) Classifier

SVM is a machine learning classifier which is based on the vector space, and it aims to determine the boundary between two or multiple output classes. SVM was first introduced by the Boser et al. [111] in 1992. In the field of information processing and computer vision, SVM gained popularity among researchers and was considered one of the most efficient and robust approaches for pattern classifications [50, 52, 60, 112]. Talking about the utility of SVM in computer vision based on drowsiness and distraction detection systems, in most of the literature presented in Chapter 2, it can be observed that SVM is one of the most often used techniques for classification, and it provided good classification results.

To perform the classification task, SVM required data sample to train itself. Once the SVM is trained, it can be applied for test data to solve the classification problem. For any basic classification problem, there are certain features based on what target values or classes are predicted. The simplest SVM classifier is called the binary classifier in which input features are predicted to be one of two output classes. For more complex classification tasks, non-linear and kernel-based SVM classifiers are available [113]. One of the most important properties of SVM is that it involves the optimization of convex function during the learning of parameters which results in no false minima. Furthermore, unlike neural networks, SVM requires only a few parameters are required for tuning and training purposes [114].

Given the case of the simplest binary classification, (x_i, y_i) is a set of input labelled points. Where, x_i is the set of feature vectors and $y_i \in \{-1, +1\}$ is class labels, assuming that both classes are linearly separable. The binary SVM classification function as represented in Equation 3.11 will be used to construct a rule in which any input feature vector x will be assigned to one of the two classes.

$$f(x; w, b) = < w, x > + b \tag{3.11}$$

Where w and b denotes the decision hyperplane vector and the intercept term, respectively. The aim of the function is to predict the output class for the input feature to be one of the two separable classes in the space. It is matter of fact that there can possibly be more than one solutions available for the decision boundary. However, SVM chooses the decision hyperplane with the maximum margin, where a margin is the minimum distance between the plane and any of the sample data point. Figure 3.9 shows a feature space for a binary classifier, in which $(w, -b)$ defines the decision hyperplane, and γ describes the margin from the decision line [115].

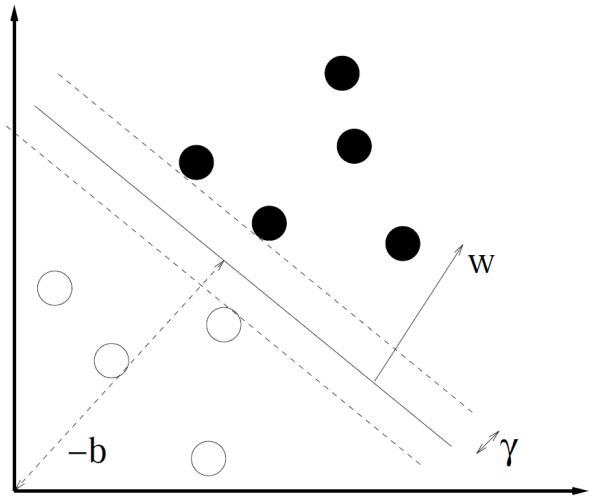


Figure 3.9: Vector Space Representation of a Linear Binary SVM Classifier [115].

The points which determine the location of the plane boundary are referred to as support vectors. Mathematically, the maximum margin solution can be formulated as the optimization problem as illustrated in Equation 3.12 and Equation 3.13.

$$\text{minimize } \frac{1}{2}||w||^2 \tag{3.12}$$

$$\text{subject to } y_i(w^T x_i + b) \geq 1, \forall i \tag{3.13}$$

SVM are based on the statistical learning theory which proves that the bounds for the generalization error can be obtained. Bounds are the function of complexity and are used for training data misclassification errors. Maximizing separation margins reduces the

overall function complexity and minimizes the bounds for generalization errors, a function that is desired in the classification problem. In simple words, maximizing separation margin results in better generalization and probability. In other words, it can be said that models with high capacity usually results in the overfitting of training data and thus in poor generalization [116].

If two classes are not linearly separable, a technique called soft margins is used to handle such problems. The idea behind soft margins is to minimize the influence of individual sample data points and allow some training points to be misclassified. To achieve this, slack variables are defined $\xi_i \geq 0$. Where $\xi_i \in \{\xi_1 \dots \xi_n\}$ is the slack variable for each data point. Hence, in this case, the mathematical formulation of the optimization problem changes as shown in Equation 3.14 and Equation 3.15

$$\text{minimize } \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i \tag{3.14}$$

$$\text{subject to } y_i(w^T x_i + b) \geq 1 - \xi_i, \xi_i \geq 0, \forall i \tag{3.15}$$

where C controls the trade-off between the minimization of error and the maximisation of plane margins.

Summary

This chapter presented a review of the conventional approaches used for detecting drowsiness and/or distraction of drivers. These conventional approaches tend to focus on a particular local region of the human body, for example the eye, the mouth and the head. Experiments has been conducted in this research to evaluate some of the commonly used conventional approaches, which is detailed in the next chapter.

CHAPTER 4

EXPERIMENTS WITH CONVENTIONAL APPROACHES

In this research study, conventional computer vision approaches that are commonly used for drowsiness and distraction detection have been experimentally tested on images to evaluate their working performance. In particular, experiments have been conducted on algorithms for detecting eye blinking, yawning and head pose, respectively. This chapter outlines the functional block diagrams of the proposed algorithms, information about image datasets, and results of conducted experiments. All the experiments were implemented using MATLAB as a software tool.

4.1. The Detection Algorithms

The implementation of each individual detection algorithm i.e. eye blinking, yawning and head pose detection is presented in the form of block diagram in Figure 4.1. All the three detection units follow the same approach and consist of two phases: training phase and testing phase. First, the SVM classifier for blinking, yawning and head pose detection is trained using a diverse set of features that are generated through the steps of detecting the facial regions, localising the facial regions of interest, and extracting the relevant features from each facial region and finally training the SVM based on those features. Once an SVM classifier is trained, the same steps can be followed for unseen test images which include detecting the facial regions, extracting the relevant features using the defined formulations and predicting the outcome based on the trained SVM classifier model. Eye blinking and yawning each has been used to assist drowsiness detection. Head pose has been used in this work to determine whether a driver is looking straight ahead (as in safe driving) or looking away/down (as in distracted or drowsy driving)

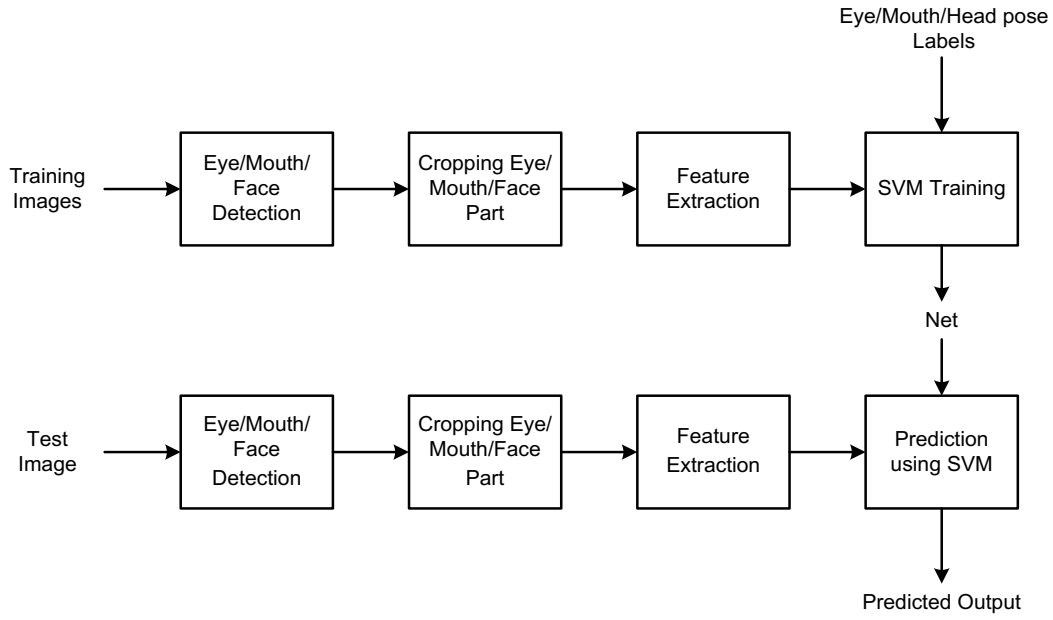


Figure 4.1: Block Diagram for the Proposed Eye Closure, Yawning and Head Pose Detection Mechanisms.

4.2. Image Datasets

The proposed conventional computer vision approaches have been trained and evaluated over three different datasets for the detection of yawning, head pose and eye blinking, respectively.

The Pointing'04 database [117] has been used for head pose training and testing purposes. The pointing'04 database consists of 2970 images captured from 15 subjects of various skin tones. The images in the dataset were captured under constant illumination conditions with white background from the constant distance of one meter to achieve face-focused image. All images in the dataset were captured in room settings, and the maximum allowed head angle was 45 degrees. From this big dataset of head pose images, in total, 200 images were randomly taken and used in our experiment to train and test the proposed head pose detection algorithm. Small set of images were used since the dataset referred to was not specifically designed for this purpose and relevant images were manually taken from the dataset and were manually labelled to be used for this experiment. Before they were

subjected to training, all the coloured images were converted to grayscale. Figure 4.2 presents some sample images from the dataset used for the head pose detection.

The training and testing of yawning detection algorithm have been performed using the Birmingham University 3D Facial Expression (BU-3DFE) dataset [118] which consists of data from 100 different subjects with 2500 different facial expressions. The dataset is diverse in terms of age ranges (18 – 70 years), skin tone and nationality. All images in this dataset include images of the face captured using front view, black background and constant illumination conditions. Furthermore, it is important to mention that all the images are captured in room settings but not in real in-vehicle settings. Figure 4.3 presents some example images from the BU-3DFE dataset.

Figure 4.2: Examples from Head Pose Dataset.

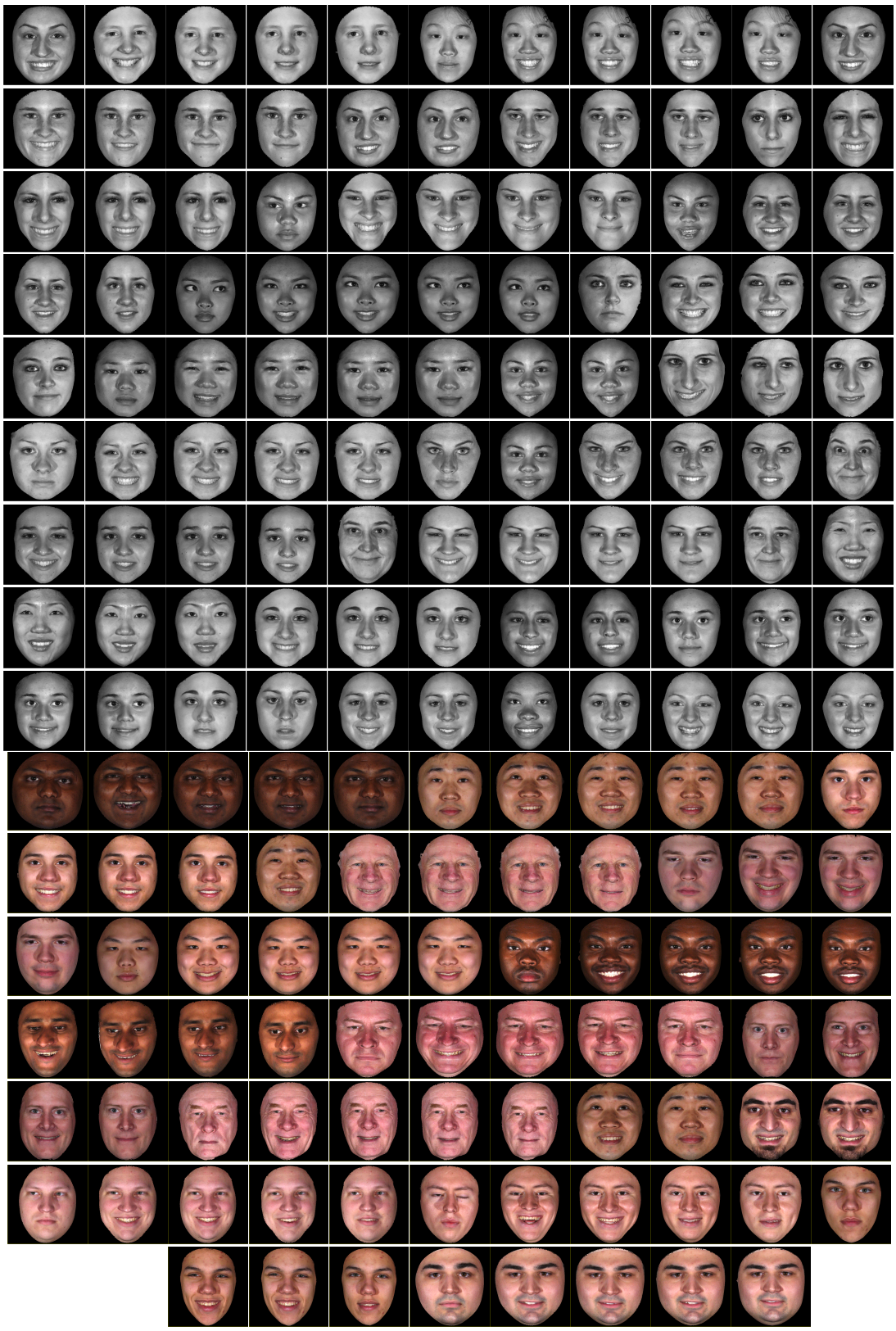


Figure 4.3: Examples from Yawning Dataset.

In general, there are a number of eye datasets and databases available publicly [119]. However, most of these benchmark and major datasets are designed for eye localization, position of eyes and coordinates of pupils. Although, many of these datasets contain a good number of eye images from a diversity of subjects, but, all the images in these datasets contain open eyes and therefore cannot be used to train and test the eye blinking classifier. For the research performed in this thesis, the dataset [120] used for the eye blinking include in total 200 images, 100 for open eyes and 100 for closed eyes. Images are colored and each contains only the cropped region of a pair of eyes. Ground truth includes the labels of open and closed eyes. The dataset containing both close and open eye images has been used to train and test the eye blinking classifier. The dataset used for our eye blinking detection experiment contains cropped colored (RGB) eye images from a diversity of subjects. Images were converted to grayscale before being subjected to the training and testing. Figure 4.4 shows few sample images used for eye blink detection algorithm.

Figure 4.4: Examples from Eye Blinking Dataset.

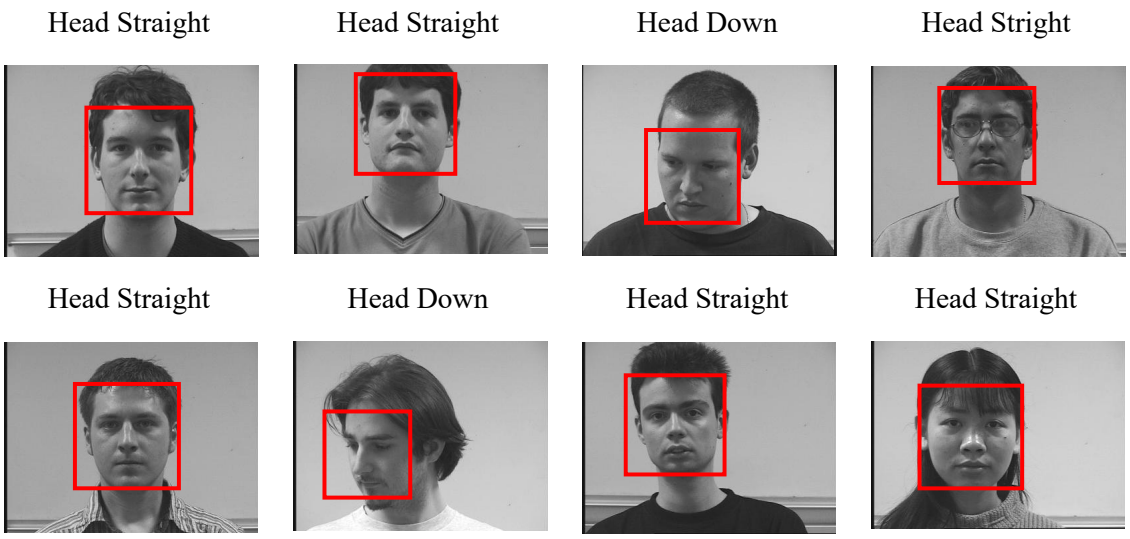
4.3. *k*-Fold Cross Validation

In order to assess the performance of proposed head pose, yawning and eye blinking algorithms for a generalized and independent dataset, *k*-fold cross validation approach has been implemented. In this approach, all the data in a dataset is randomly divided into k batches or bins of images. During each run of *k*-fold cross validation, one batch is used for testing and the rest for training. This process keeps on iterating for all the batches. In all

our experiments the value of k is selected as 10. In the end, results of each batch were then averaged to get generalized precision of the correct and false detection of the proposed algorithm.

4.4. Head Pose Detection Results

The head pose detection classifier was trained with the images in the Pointing'04 database. Labels used by the SVM classifier for prediction are binary in the sense that the face in the image is either straight or tilted. It is important to mention that the dataset was manually labelled in this study. Figure 4.5 (a) and Figure 4.5 (b) present examples of correct and wrong pose detection cases with the face enclosed in a rectangular region along with SVM predicted labels, respectively.



(a) Correct Detections

(b) Wrong Detections

Figure 4.5: (a) Examples of Correct Head Pose Detection (b) Examples of Wrong Head Pose Detection.

Table 4.1 presents the detailed results of the 10-fold cross validation experiments. In total, 10 iterations were performed and in the end detection accuracies were averaged. Overall, the head pose detection algorithm showed a correct detection accuracy of 98.50% and false detection of 1.5%. High classification accuracies of SVM are because a controlled and relatively small dataset with not much of diversity has been used. The few wrong classifications may be due to the small training set used in the experiments.

Table 4.1: k-Fold Cross Validation Results of Head Pose Detection Algorithm.

Number of Iteration (k)	**Accuracy detection(%)**	**False detection(%)**
1	100.0	0.0
2	100.0	0.0
3	100.0	0.0
4	90.0	10.0
5	100.0	0.0
6	95.0	5.0
7	100.0	0.0
8	100.0	0.0
9	100.0	0.0
10	100.0	0.0
Average	**98.50**	**1.5**

In Figure 4.6 as shown below, the first two diagonal cells show the number and percentage of correct classifications by the trained network. As shown, 98 images from the class belonging to ‘Head Straight’ are correctly classified. This corresponds to 98.0% of all the100 images in the class belonging to ‘Head Straight’. Similarly, 99 images are correctly classified from the class belonging to ‘Head Down’. This corresponds to 99.0% of all the100 images in the class belonging to ‘Head Down’. Two (2) images belonging to the class of ‘Head Straight’ are incorrectly classified and this corresponds to 2.0% of all 100 images in the class belonging to ‘Head Straight’. Similarly, 1 image belonging to the class

of ‘Head Down’ is incorrectly classified and this corresponds to 1.0% of all 100 images in the class belonging to ‘Head Down’. After applying the k-fold cross validation, Overall, 98.5% of the predictions are correct and 1.5% are wrong

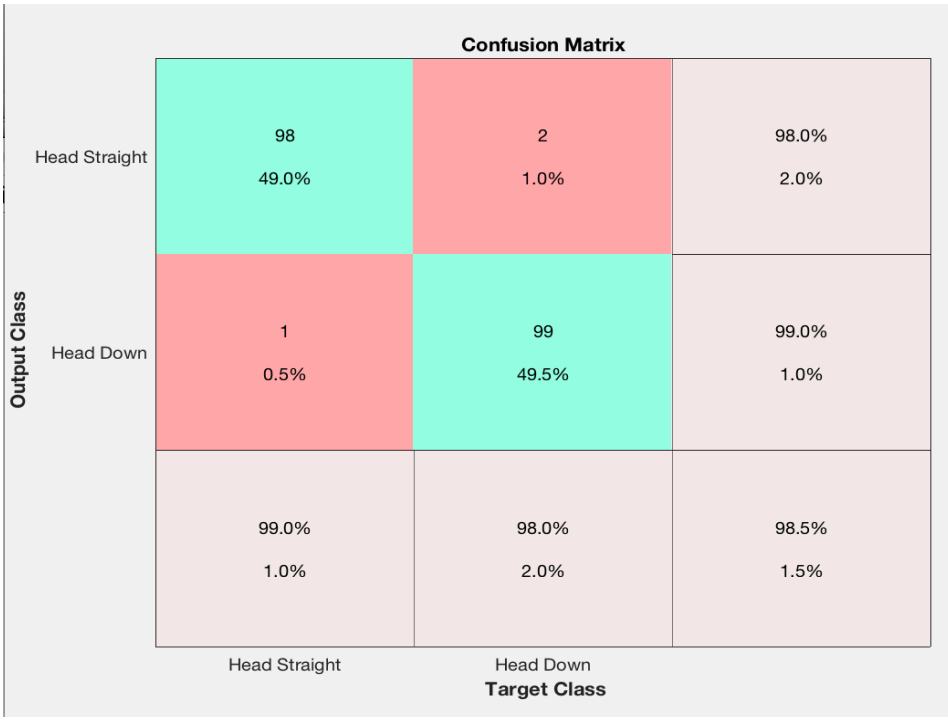
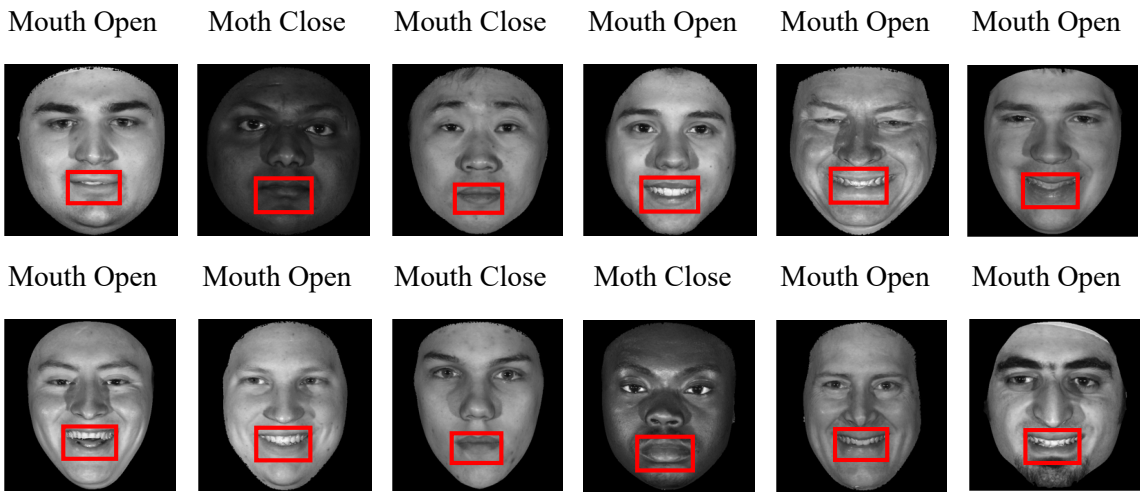


Figure 4.6: Confusion Matrices For k-Fold Cross Validation Results Head Pose Detection

4.5. Yawning Detection Results

Yawning detection using the DOO features defined in equation 3.8 has exhibited promising results despite the relatively small dataset used. It is important to mention that the ground truth labelling was not available within the dataset and the dataset was therefore manually labelled. As a result, the labelling was subjective in nature. Figure 4.7 (a) and Figure 4.7 (b) present the instances of correct and wrong detections of yawning detection, including the predicted labels of mouth opened and mouth closed. Furthermore, detected mouths are enclosed in a rectangle region.



(a) Correct Detections

(b) Wrong Detection

Figure 4.7: (a) Examples of Correct Yawning Detection (b) Examples of Wrong Yawning Detection.

Table 4.2 presents the details of k-fold cross validation applied to assess the performance of yawning detection algorithm. Overall, on average, 99.0% correct classification accuracy and 1.0% false classification accuracy was achieved from the cross validation.

In Figure 4.8, as shown below, the first two diagonal cells show the number and percentage of correct classifications by the trained network. As shown, 100 images from the class belonging to 'Mouth Open' are correctly classified. This corresponds to 100.0% of all the100 images in the class belonging to 'Mouth Open'. Similarly, 98 images are correctly classified from the class belonging to 'Mouth Close'. This corresponds to 98.0% of all the 100 images in the class belonging to 'Mouth Close' There is no incorrect classification of image in the class belonging to 'Mouth Open'. Similarly, Two(2) images belonging to the class of 'Mouth Close' are incorrectly classified and this corresponds to 2.0% of all 100

images in the class belonging to 'Mouth close'. After applying the k-fold cross validation, Overall, 99.0% of the predictions are correct and 1.0% are wrong

Table 4.2: k-Fold Cross Validation Results of Yawning Detection Algorithm.

Number of Iteration (k)	**Accuracy detection(%)**	**False detection(%)**
1	100.0	0.0
2	100.0	0.0
3	100.0	0.0
4	100.0	0.0
5	100.0	0.0
6	95.0	5.0
7	100.0	0.0
8	100.0	0.0
9	95.0	5.0
10	100.0	0.0
Average	**99.0**	**1.0**

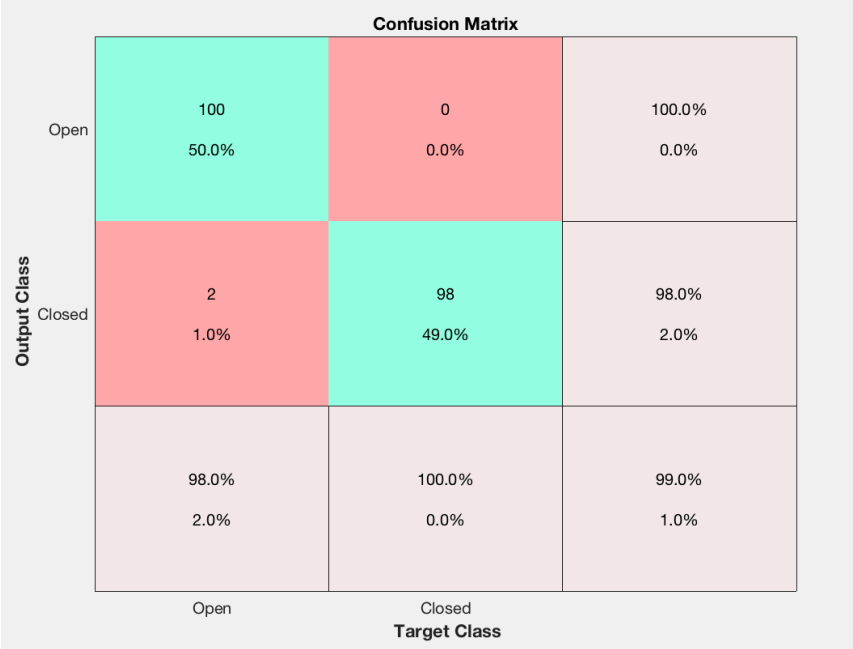
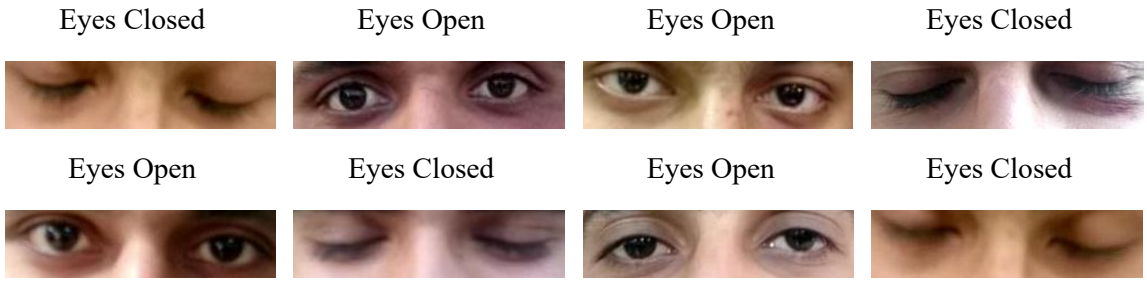


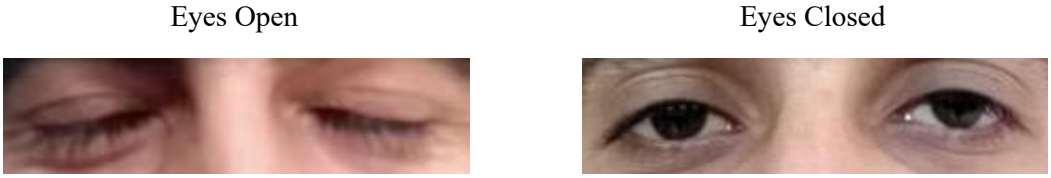
Figure 4.8: Confusion Matrices For k-Fold Cross Validation Results Yawning Detection.

4.6. Eye Blink Detection Results

For eye blinking detection, Viola and Jones algorithm and Haar-like features have been used in our experiments. Manually labelled groundtruth was provided with the dataset. Figure 4.9 (a) and Figure 4.9 (b) present examples of some correct and false cases of eye blink detection along with SVM predicted labels, respectively.



(a) Correct Detection



(b) Wrong Detections

Figure 4.9: (a) Examples of Correct Eye Blink Detection (b) Examples of Wrong Eye Blink Detection.

Table 4.3 presents the 10-fold cross validation results for the eye blinking classifier. Overall, an accuracy of 99.0% has been achieved. High detection accuracies of SVM are because of similar samples of data in the testing and training datasets.

In Figure 4.10, as shown below, the first two diagonal cells show the number and percentage of correct classifications by the trained network. As shown, 99 images from the class belonging to ‘open eye’ are correctly classified. This corresponds to 99.0% of all the100 images in the class belonging to ‘open eyes’. Similarly, 99 images are correctly classified from the class belonging to ‘closed eye’. This corresponds to 99.0% of all the 100 images in the class belonging to ‘closed eyes’ 1 image belonging to the class of open eye is incorrectly classified and this corresponds to 1.0% of all 100 images in the class belonging to ‘open eyes’. Similarly, 1 image belonging to the class of ‘closed eye’ is

incorrectly classified and this corresponds to 1.0% of all 100 images in the class belonging to 'close eyes'. After applying the *k*-fold cross validation, Overall, 99.0% of the predictions are correct and 1% are wrong

Table 4.3: *k*-Fold Cross Validation Results of Eye Blink Detection Algorithm.

Number of Iteration (K)	Accuracy detection(%)	False detection(%)
1	100.0	0.0
2	100.0	0.0
3	100.0	0.0
4	100.0	0.0
5	100.0	0.0
6	90.0	10.0
7	100.0	0.0
8	100.0	0.0
9	100.0	0.0
10	100.0	0.0
Average	**99.0**	**1.0**

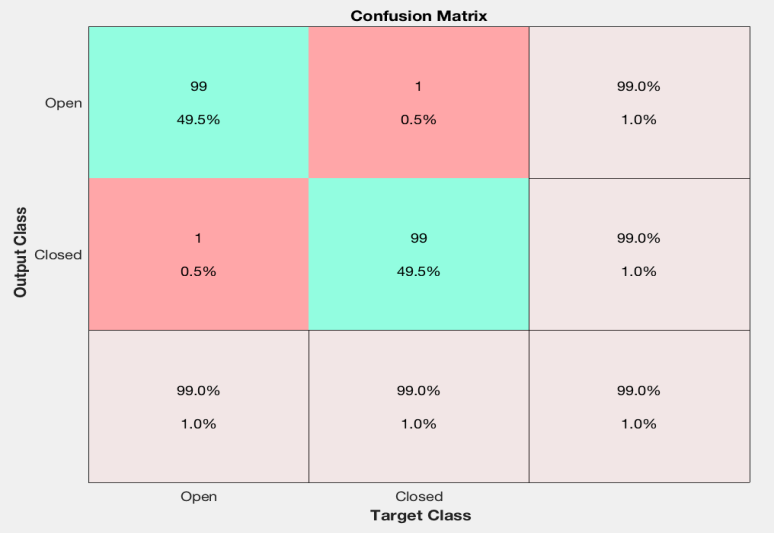


Figure 4.10: Confusion Matrices For *k*-Fold Cross Validation Results Eye Blinking Detection.

Table 4.4 presents the overall statistics of SVM classification for drowsiness and distraction related features i.e. yawing, head pose and eye blinking. In terms of number of images, from total of 200 images in each case, head pose detection and eye blinking detection algorithms were able to correctly classify 198 images while 2 images were incorrectly classified. On the other hand, for the head pose detection, out of 200, 197 were correctly classifier where 3 were incorrectly classified. Overall, accuracy of 98.83% has been achieved using SVM classifier. Few incorrect instances were because of reasons presented in head pose results section.

Table 4.4: Statistics of SVM Classification for Drowsiness Detection.

	Number of Images	**Correct Classification**	**False Classification**	**Classification Accuracy**
Yawning	200	198	2	99%
Head Pose	200	197	3	98.5%
Eye Blinking	200	198	2	99%
Overall	**600**	**593**	**7**	**98.83%**

The proposed drowsiness and distraction system based on the conventional computer vision approaches has performed satisfactorily in terms of classifying the drowsiness-related behaviours. In this research, the main purpose of performing these experiments was to investigate the effectiveness of the conventional computer vision approaches in terms of drowsiness detection. Nevertheless, in order to make the proposed algorithms robust and practically functional, certain limitations still need to be addressed. The algorithms were not trained and tested with comprehensive datasets such as the Kaggle dataset that will be used in Chapter 6 of this thesis; rather, relatively small datasets were used to train an SVM as well as evaluate its performance. This means that the reported performance figures might not be the same when the trained classifier is applied to images of other unseen subjects. Furthermore, the datasets used for the experiments were captured in room-settings rather than in vehicle real-time settings. Thus, it cannot be proven that the tested algorithms will work with similar accuracies with images such as those in the Kaggle dataset. To work with the Kaggle dataset and perform classification of various driving behaviors, Chapter 5

investigates the feasibility of deep learning architectures and Chapter 6 shows their experimental results on the Kaggle dataset.

Chapter 5

DEEP LEARNING APPROACHES

5.1. Introduction

To overcome the drawbacks of conventional vision approaches in terms of accuracy and generalization identified from the experiments reported in previous chapters, deep learning approach-based techniques have been proposed and implemented. Deep learning approaches such as CNN have proved to improve the classification accuracies and generalization issues. The second half of this MPhil study attempted to implement different configurations of deep CNN architectures and compared their performance for a Kaggle challenge of detecting distracted drivers. A number of deep architectures such as AlexNet, ResNet, MobileNet and NASNet have been implemented to compare the performance using Softmax and SVM classifiers with cross-entropy loss and hinge-loss respectively. Experimental design and results are presented in the next chapter. This chapter explains the CNN architectures, which are one of the deep architectures commonly used for computer vision tasks. Theoretical details about different CNN architectures (such as AlexNet, ResNet, MobileNet and NASNet) which are used in this research have been included. Finally, this chapter provides the information about the triplet loss and batch triplet-loss functions.

5.2. Deep CNN Architectures

This section provides the details about four different deep architectures AlexNet, ResNet, MobileNet and NASNet, all of which are implemented in this research to classify the distracted driving behaviours. This section provides the theoretical basis to each of these deep networks.

5.2.1. AlexNet

Krizhevesky et al. [74] proposed a deep CNN architecture named AlexNet as shown in Figure 5.1. AlexNet was the winner of one of the most advanced and challenging object recognition competitions called ImageNet [121], which involves the classification of the

real-life images into one of 1000 prediction classes. AlexNet was trained over 1.3 million images from the ImageNet challenge database and achieved the test error rate of 15.3% [74]. First five layers of the model are convolution layers. Also used are max pooling layers with Local Response Normalization (LRN) and 96 different filters of 11×11 size. For the max pooling operation, a filter size of 3×3 with the stride of 2 is used. The same operations are performed in the second layer with 5×5 filters. A filter size 3×3 is used in the third, fourth, and fifth convolutional layers with 384, 384, and 296 feature maps respectively. Two fully connected (FC) layers are used with dropout followed by a Softmax layer at the end.

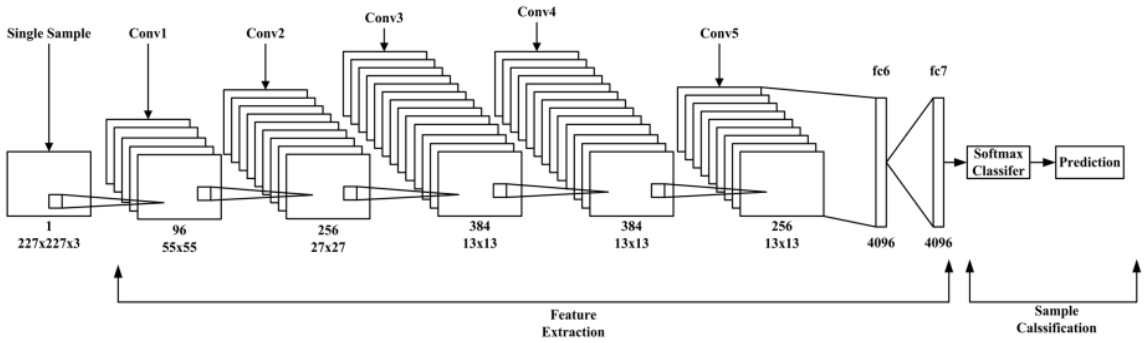


Figure 5.1: Structure of the Adopted AlexNet Deep Architecture used in this Research.

Networks with similar structure and the same number of feature maps are trained in parallel for this model. Two new concepts, LRN and dropout, are introduced in this network. LRN can be applied in two different ways: first, it can be applied on a single channel or feature maps, where an $N \times N$ patch is selected from the same feature map and normalized based one the neighbourhood values. Second, LRN can be applied across the channels or feature maps (neighbourhood along the third dimension but a single pixel or location).

AlexNet network has been chosen for this research because this deep architecture has already demonstrated its ability to classify objects in ImageNet challenge over large number of images. Besides, features that are already learnt by this network are also useful in detecting driver distraction. Hence, rather than training the architecture from scratch, preserving the features learnt from ImageNet challenge will fasten the training process and reduce the risk of overfitting. From the previously learn features, AlexNet will be able to

classify objects such as phones, pets, hand, cups, and coke cans, all of which are valuable measures in classifying distracted driving.

AlexNet has 5 convolution layers and 2 fully connected layers. When processing the ImageNet dataset, the total number of parameters for AlexNet can be calculated as follows for the first layer: input samples are $224 \times 224 \times 3$, filters (kernels or masks) or receptive field have a size of 11, the stride is 4, and the output of the first convolution layer is $55 \times 55 \times 96$. We can calculate that this first layer has 290400 ($55 \times 55 \times 96$) neurons and 364 ($11 \times 11 \times 3 = 363 + 1$ bias) weights. The parameters for the first convolution layer are $290400 \times 364 = 105{,}705{,}600$. The total number of weights and Multiply and Accumulated (MACs) for the whole network are 61M and 724M, respectively.

The modified architecture of the AlexNet deep network for Kaggle challenge is explained as follows. Each input is of the size defined by Kaggle challenge i.e. $227 \times 227 \times 3$. The first five layers in the architecture are local layers, and they provide the representation of local features, while last layers are responsible for learning and classification of features and are fully connected layers. At the layer fc7, a total of 4096 features are extracted and saved in a matrix $\mathbf{X}$. The dimension of feature matrix $\mathbf{X}$ is $m \times 4096$, where m is the number of training images in each batch. In this case, m equals 50. At the next stage, the Softmax classifier is given with this feature matrix, which then classifies those features in one of the 10 Kaggle classes. The output probability values from the Softmax classifier are compared with the ground truth labels to calculate classification loss.

5.2.2. ResNet

Proposed by Kaiming He, ResNet [75] was named after the residual connection which is a connection between convolution layers as shown in Figure 5.2. ResNet was proposed with the aim of developing an ultra-deep network towards the solution of gradient problem in the conventional deep CNN architectures.

ResNet was proposed to have different number of layers i.e. 34, 50, 101 and 152 depending on the type of application used. The most popular of them was ResNet50 with 50 layers deep, 49 convolutional layers and 1 fully connected layer at the end of the network. The

total number of weights and MACs for the whole network are 25.5M and 3.9G, respectively. Figure 5.3 presents the architecture of the ResNet proposed by author of [75] for ImageNet challenge.

ResNet is a traditional feed forward network with a residual connection. The output of a residual layer can be defined based on the outputs of $(l-1)^{th}$ which comes from the previous layer defined as x_{l-1}. $F(x_{l-1})$ is the output after performing various operations, e.g., convolution with different sizes of filters, Batch Normalization (BN), and an activation function such as ReLU on x_{l-1}. The final output of residual unit is x_l which can be defined with Equation 5.1.

$$x_l = F(x_{l-1}) + x_{l-1} \tag{5.1}$$

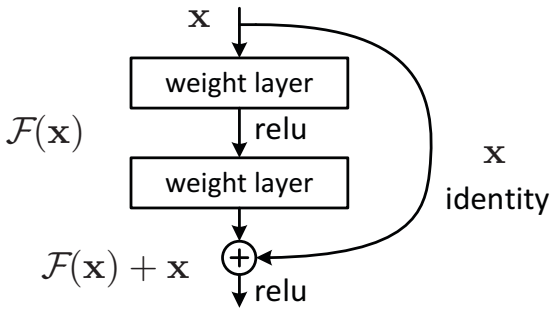


Figure 5.2: Residual Learning, Building Block of ResNet (Taken From [75]).

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 5.3: Architecture of ResNet Proposed for the ImageNet Challenge (Taken From [75]).

The residual network consists of several basic residual blocks. However, the operations in the residual block can be varied depending on the different architectures of residual networks. The wider version of the residual network was proposed by Zagoruvko et al. [122]. Another improved residual network approach known as aggregated residual transformation was proposed in 2016 [123]. Recently, some other variants of residual models have been proposed based on the residual network architecture [124-126].

5.2.3. MobileNet

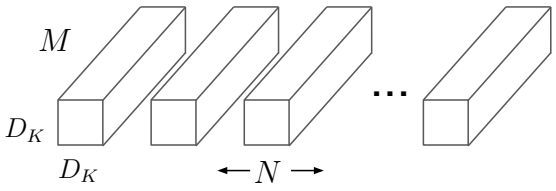
In the deep learning domain, trends are shifted to develop more deeper and complex architectures and to improve the performance of the tasks [127-129]. However, making more deeper architectures was not always efficient in terms of size and time. In most real-world applications, the task is to perform the classification quickly using the limited computational resources (on board processing units); therefore, complex and deeper networks were proving not to be the real-world architectures.

In an effort to achieve the improved performance by not using deeper layered architecture, Howard et al. [130] proposed MobileNet, a fast, precise and most importantly a low size network. The concept behind the MobileNet was the use of depthwise convolutions which are a form of factorized convolutions. In simple words, standard convolution operation was factorized into 1×1 pointwise convolutions. Factorizing the standard convolutions into pointwise depth convolutions significantly reduced the size of the model and increase the processing speed [130]. Figure 5.4 presents the illustration of standard convolution, depthwise convolution and pointwise depth convolution.

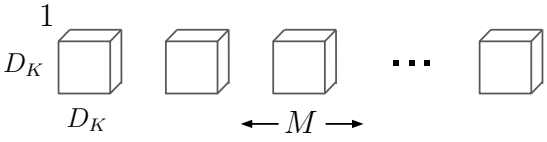
A standard convolution layer takes a $D_F \times D_F \times M$ feature map $\mathbf{F}$ as the input to the layer and produces a $D_G \times D_G \times N$ feature map $\mathbf{G}$. D_F denotes the spatial height and width of the squared input while M represents the number of input channels. Similarly, D_G denotes the spatial height and width of the output feature map and N represents the number of output channels. Mathematically, the output of a standard convolution can be expressed as shown in Equation 5.2.

$$\mathbf{G}_{k,l,n} = \sum_{i,j,m} \mathbf{K}_{i,j,m,n} \cdot \mathbf{F}_{k+i-1,l+j-1,m} \tag{5.2}$$

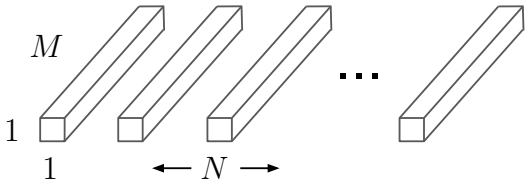
where **F** denotes the input feature map, **K** represents the kernel size and **G** denotes the output feature map.



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Figure 5.4: Concept of Depthwise Convolution Proposed by Howard et al. [130].

Unlike the standard convolution, depthwise separable convolution operation involves two layers: a depthwise convolution and a pointwise convolution. MobileNet uses depthwise convolution operations by first applying a single filter to each input depth and then applying 1×1 pointwise convolution to produce linear combination of depthwise outputs. Both ReLU and BatchNorm non-linearites are used in MobileNet. This combination of depthwise convolution and pointwise convolution is referred to as the depthwise separable convolution with the computational cost as expressed in Equation 5.3.

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F \tag{5.3}$$

Expressing convolution operation as a two-step filtering process can be reduced as expressed in Equation 5.4.

$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F} = \frac{1}{N} + \frac{1}{D_K^2} \quad (5.4)$$

Figure 5.5 shows the architecture of MobileNet CNN Model as proposed by Howard et al. [130].

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1 Conv / s1	$3 \times 3 \times 512$ dw $1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$ $14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Figure 5.5: Architecture of MobileNet CNN Proposed by Howard et al. [130].

5.2.4. NASNet

The development of deep neural networks for the tasks of image classification is based mainly on the manual or human architecture engineering. With better architecture, improved results have been achieved in the past over a number of classification tasks. However, Zoph et al. [131] proposed a new paradigm for deep neural network development by introducing the idea of using a scalable method to optimize the deep architecture for the dataset where it is intended to be applied. The proposed method for scalable optimization is based on the Neural Architecture Search (NAS) which uses reinforcement learning approach to optimize the architecture.

The implementations of the NAS or any other search method for huge image datasets is challenging in terms of computational power. Because of this challenge, Zoph et al. [131] proposed the idea of optimizing the good architecture over proxy dataset (small) and then transferring the learned architecture to the actual dataset (large). This transfer of learning

is achieved by using the NASNet search space, and that is why this type of architectures is known as NASNets. The NAS search uses the Recurrent Neural Network (RNN) to sample the child networks [126]. Sampling is done with a probability p. Once the architecture is sampled, it gets trained on the specific dataset. When the training phase reaches a local minimum and an accuracy R, gradients of p gets scaled by R, the RNN controller gets updated and better architectures are achieved over time. Figure 5.6 shows the function block diagram of the RNN controlled based NAS.

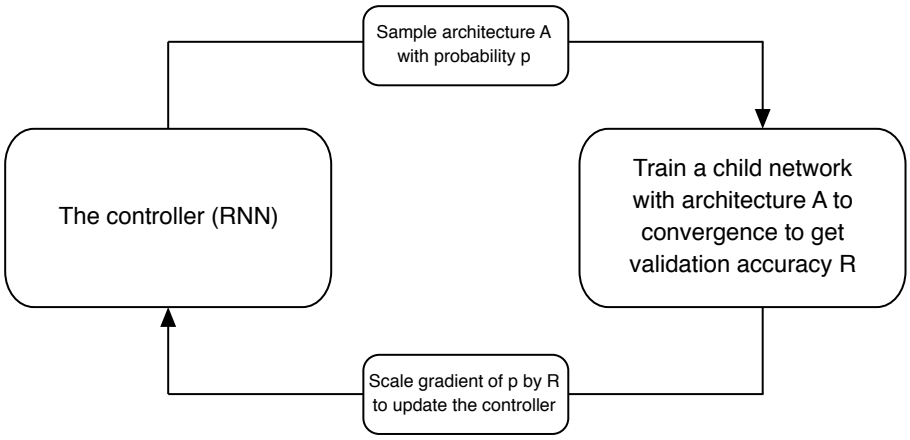


Figure 5.6: Overview of Neural Architecture Search (NAS) (Taken From [126]).

5.3. Theory of Transfer Learning

Transfer learning is the approach in machine learning where well-trained models for one problem are tuned and used for different but similar problems. Because training a deep architecture from scratch requires huge dataset, computational resources and time, transfer learning concept has gained popularity among researchers and provided reasonable results when practiced in real-world problems. In technical terms, when a model is trained over comprehensive dataset, it gains the knowledge and stores it in the form of layer weights. In transfer learning approach, pre-trained models are used while preserving the weights learned for previous problem and then fine-tuned to conform to the requirement of the new problem. For example, a deep network learned over huge dataset of ImageNet challenge will have features learned, and they are useful for most object-detection and classification problems in computer vision. Hence, if these features are used with fine-tuning, they will

perform effectively for new problems. There are two commonly used approaches in transfer learning. In the first approach, new layers are added over the previously learned layers, and the whole architecture is trained including the previously learned weights. However, this approach results in breaking of previously learned features if the learning rate is not selected carefully. In the second approach, only the newly added layers in the architecture are learned while preserving the weights from the pre-learned model. In practice, a combination of both approaches is more commonly used and proved effective [132].

In other words, the usual transfer learning approach is to train a base network and then copy its first n layers to the first n layers of a target network. The remaining layers of the target network are then randomly initialized and trained for the target task. It is possible to choose to back-propagate the errors from the new task into the base features and fine-tune them to the new task. Besides, the transferred feature layers can be left frozen, but leaving them frozen usually turns out to be the best choice.

5.4. Distance Metric Learning

Metric learning is a long-standing problem whose main goal is to find a function $f(\boldsymbol{x}_i, \boldsymbol{x}_j)$ that gives us a distance between items $\boldsymbol{x}_i$. The learned distance function is problem dependent, symmetric $f(\boldsymbol{x}_i, \boldsymbol{x}_j) = f(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and non-negative $f(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$. In particular, we will focus on how to learn the distance function f over images using CNN [133-135].

With the recent computational capabilities and the increase in the data available online, CNNs have become a de facto standard model for tasks involving image manipulation due to its great performance. The concatenation of linear and non-linear layers squeezes the raw information given by image pixels. Together with the back-propagation algorithm [136], they build a high-level feature representation of the input that is encoded into a feature vector that has shown to have outstanding capabilities [137-140]. These feature vectors are usually the last layers of the model, and they encode the high and low-level information in the network extracted from the input image. In other words, the CNN can be regarded as a feature extractor that maps an input xi into a new Euclidean space $M(x_i) \in \mathbb{R}^d$ where d is the dimensionality of the resulting feature vector.

5.5. Triplet Loss

Distance metric learning is the approach that improves the classification efficiency for real-world applications and aims to keep the data points from the same classes close to each other and to ensure data points from the different classes are far apart from each other. At the fully connected layer in a CNN, the output is in the form of feature representations vector which are then classified by Softmax or Support Vector Classifier (SVC). However, it has been observed that if the feature vectors are learned in a way that information is spatially separated and features are embedded into a new space where the position of features is an indication of the similarity with certain class, the classification accuracy of the overall network will improve. Furthermore, the reduced dimensionality of the new space allows the classifier model to learn good boundaries for classes more quickly.

Triplet loss is the approach in distance metric learning which is used to train the CNN in a way that it maps feature representations to a d-dimensional Euclidean space $\left(f(\boldsymbol{x}) \in \mathbb{R}^d, \left|\left|f(\boldsymbol{x})\right|\right|_2 = 1\right)$ where the position of the feature representations indicates the similarity to certain class [77]. Unlike categorical cross-entropy, optimizing triplet loss does not directly involve improving the class-prediction accuracy; rather, it focuses on the creation of an encoding in which different classes are separated spatially. Figure 5.7 presents the three main components involved in triplet loss: positive, anchor and negative. Triplet loss function aims to improve the classification accuracy by increasing the distance between the anchor and the negative while decreasing the distance between anchor and negative during the learning process. The triplet of samples means the anchor and positive samples belongs to the same class, while negative sample belongs to a different class. Optimizing the loss function causes the CNN to bring the encoding for the anchor and positive example closer together and to separate the anchor and negative sample. This is done for an entire batch of triplets at a time. In this way, the model learns to map related data to a similar position in the encoding space, while separating unrelated data [141, 142].

5.6. Activation Functions with Triplet Loss

Knowing that CNNs are powerful feature extractors, we can find the function $f(M(\boldsymbol{x}_i), M(\boldsymbol{x}_j))$ that will give us the distance between two input images $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. In an ideal case, we would like both images to be close in the Euclidean space when both images belong to the same semantic class $f(M(\boldsymbol{x}_i), M(\boldsymbol{x}_j)) \simeq 0$. We would also like to have a large distance between them when they do not belong to the same class $f(M(\boldsymbol{x}_i), M(\boldsymbol{x}_j)) \gg 0$ [77]. In our case, f will be the Euclidean distance between feature vectors. Thus, the equation is as follows

$$f(M(\boldsymbol{x}_i), M(\boldsymbol{x}_j)) = ||M(\boldsymbol{x}_i) - M(\boldsymbol{x}_j)||_2^2$$

Moreover, the feature vectors lie on the d-dimensional hypersphere $||M(\boldsymbol{x}_i)||_2 = 1$, thus making the distance between feature vectors proportional to the cosine similarity. Using f, we can compare the distances between the anchor image $(\boldsymbol{x}_i^a)$ and the positive sample $(\boldsymbol{x}_i^p)$given by $f(M(\boldsymbol{x}_i^a), M(\boldsymbol{x}_i^p)) \simeq 0$, including the distances between the anchor and the negative sample $\boldsymbol{x}_i^n$ given by $f(M(\boldsymbol{x}_i^a), M(\boldsymbol{x}_i^p)) \gg 0$. Figure 5.7 shows the geometrical interpretation of the training performed and compares the distances with a triplet loss using a margin. To learn how to place the images of the same semantic class nearby in the Euclidean space while separating the images of different class, we use a triplet loss function involving the anchor, positive and negative images.

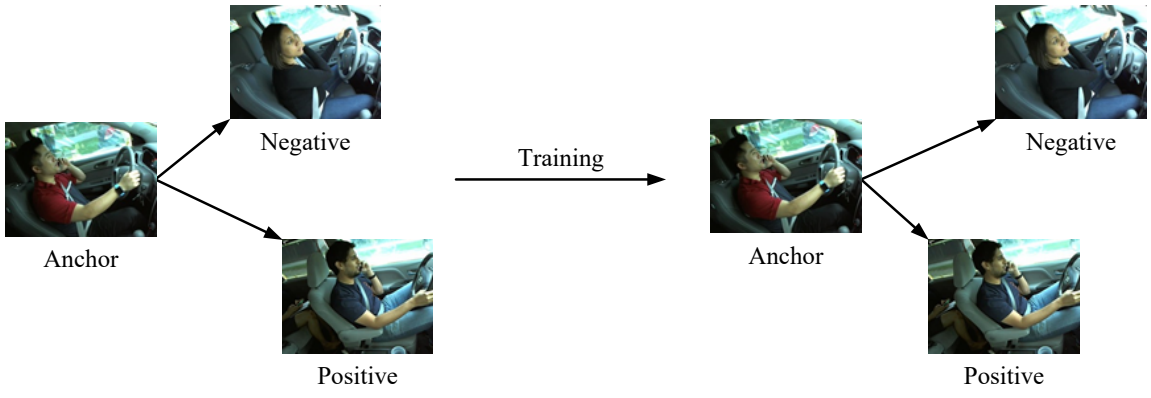


Figure 5.7: Example of Triplet Before and After Training to Illustrate the Advantage of Triplet Loss Function.

There are number of ways in which loss function, sample selection and batching are implemented using the triplet loss. Some commonly used triplet loss implementations are discussed as follows.

5.6.1. Margin Triplet Loss

The margin triplet loss function uses the simple idea that the difference between the anchor-positive vector distance and anchor-negative vector distance must be separated by a minimum margin of α [143]. Mathematically, the standard margin triplet loss function can be expressed as shown in Equation 5.5.

$$L = \sum_{i}^{N} \max(0, f(\boldsymbol{x}_i^a, \boldsymbol{x}_i^p) - f(\boldsymbol{x}_i^a, \boldsymbol{x}_i^n) + \alpha) \tag{5.5}$$

Where $\boldsymbol{x}_i^a$ denotes the anchor feature vector, $\boldsymbol{x}_i^p$ denotes the positive feature vector, $\boldsymbol{x}_i^n$ denotes the negative feature vector, and α denotes the forced margin between the anchor-to-positive distance and the anchor-to-negative distance. In Equation 5.32, the function $f(\boldsymbol{x}_i^a, \boldsymbol{x}_i^p)$ determines the distance between two feature vectors (in this case anchor and positive). From this equation, it can be observed that triplet loss function tries to separate the positive and negative samples by a margin of α. The only condition at which the triplet loss will be greater than zero is this: $f(\boldsymbol{x}_i^a, \boldsymbol{x}_i^p) + \alpha > f(\boldsymbol{x}_i^a, \boldsymbol{x}_i^n)$. This loss function is formulated in a similar way to hinge loss in that a sample only contributes to the loss and the gradient of the model if the total loss for a specific sample is positive. Additionally, a margin α is used to specify a minimum distance which the Euclidean distance squared between the embedding for $\boldsymbol{x}^a$ and $\boldsymbol{x}^p$ must reach before the loss becomes zero.

Triplet loss function can equivalently be expressed as shown in Equation 5.6.

$$\max(0, \mu_{ap} - \mu_{an} + \alpha) \tag{5.6}$$

where

$$\mu_{ap} = \frac{1}{N} \sum_{i}^{N} f(\boldsymbol{x}_i^a, \boldsymbol{x}_i^p)$$

$$\mu_{an} = \frac{1}{N}\sum_{i}^{N} f(\boldsymbol{x}_i^a, \boldsymbol{x}_i^n)$$

5.6.2. Naïve Triplet Loss

The naïve triplet loss function is similar to the margin triplet loss function. The only difference between them is that in the naïve triplet loss function, the mean operation is placed around the squared Euclidean distances. In simple words, Euclidean distances between anchor-positive and anchor-negative vectors are averaged before the margin α is applied. This means that individual samples cannot be set to zero if they are not positive. The entire batch must satisfy the margin constraint in order to set the gradient to zero. Mathematically, this can be expressed as shown in Equation 5.7.

$$L = \left[\frac{1}{N}\sum_{i=0}^{N}\left(\left\|f(\boldsymbol{x}_i^a) - f\left(\boldsymbol{x}_i^p\right)\right\|_2^2\right) - \frac{1}{N}\sum_{i=0}^{N}(\|f(\boldsymbol{x}_i^a) - f(\boldsymbol{x}_i^n)\|_2^2) + \alpha\right]_+ \quad (5.7)$$

5.6.3. Batch Triplet Loss

The third formulation of loss implemented and tested as a part of this project is the batch triplet loss.

$$d_{ap} = \left\|f(\boldsymbol{x}_i^a) - f\left(\boldsymbol{x}_i^p\right)\right\|_2^2$$

$$d_{an} = \|f(\boldsymbol{x}_i^a) - f(\boldsymbol{x}_i^n)\|_2^2$$

For batch triplet loss, the distances between embeddings for the anchor and positive anchor are calculated using squared $\boldsymbol{L}_2$ (Euclidean) distance. The same is done for the distance between the anchor and negative.

$$\mu_{ap} = \frac{1}{N}\sum_{i=0}^{N} d_{ap}$$

$$\mu_{an} = \frac{1}{N}\sum_{i=0}^{N} d_{an}$$

As for triplet loss function implementation, the separation of mean value for each distribution does not always result in improved performance as there may be overlapping

between the distributions which may result in an increased number of false positives and false negatives. In this case, for the means of distributions μ_1 and μ_2, there will be a decision threshold which will define how both distributions are well-separated. Parameters $\sigma_1$and σ_2 are the standard deviations of two distributions, and the decision overlapping can de represented mathematically as given in Equation 5.8 [144]. Figure 5.8 presents the illustration of overlapping problem graphically.

$$\text{decison threshold} = \frac{|\mu_1 - \mu_2|}{\sqrt{\sigma_1^2 + \sigma_2^2/2}} \tag{5.8}$$

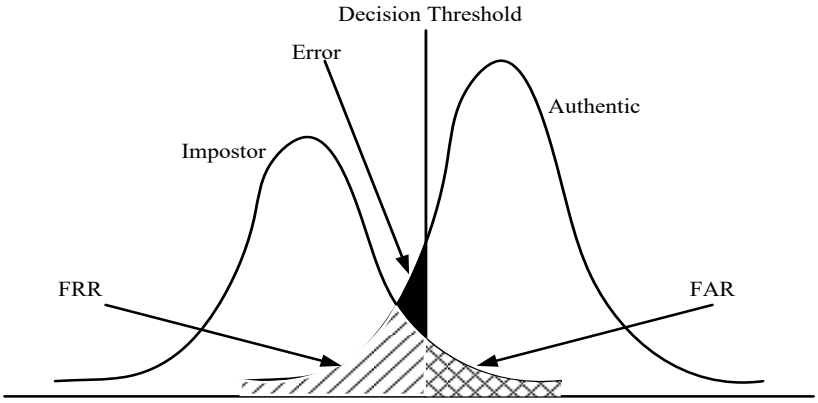


Figure 5.8: Illustration of Overlapping Issue in Triplet Loss [144].

Towards the solution of this overlapping problem, the batch triplet loss function uses the variances of the squared Euclidean distance to be minimized along with means. Equation 5.9 presents the mathematical expression for the batch triplet loss function.

$$(1-\beta)\max\left(0, \mu_{ap} - \mu_{an} + \alpha\right) + \beta\left(\sigma_{ap}^2 + \sigma_{an}^2\right) \tag{5.9}$$

Where

$$\sigma_{ap}^2 = \frac{1}{N}\sum_{i}^{N}\left(f\left(\boldsymbol{x}_i^a, \boldsymbol{x}_i^p\right) - \mu_{ap}\right)^2$$

$$\sigma_{an}^2 = \frac{1}{N}\sum_{i}^{N}\left(f\left(\boldsymbol{x}_i^a, \boldsymbol{x}_i^n\right) - \mu_{an}\right)^2$$

The term β in Equation 5.36 balances each term contribution in the loss function. At unity value of β, it can be seen from Equation 5.36 that the first term will be zero. This means that only the effect of variances will be effective for loss function. On the other hand, by selecting β as zero, the effect of variances can be eliminated from the loss function [145, 146].

5.7. Triplet Mining

Triplet mining is the process of selecting the efficient triplets from the large number of available ones. From the definition of triplet loss, triplets can be categorized into three-fold:

- Easy Triplets: This is triplets with zero loss i.e. the distance between the anchor and the negative (d_{an}) is greater than the distance between the anchor and the positive (d_{ap}) plus the margin (α). Mathematically, it can be expresses as $d_{ap} + \alpha < d_{an}$.
- Hard Triplets: This is triplets in which the distance between the negative and the anchor is less than distance between the positive and the anchor $(d_{an} < d_{ap})$.
- Semi-Hard Triplets: This is triplets in which d_{ap} is greater than d_{an} but still the loss is positive ($d_{ap} < d_{an} < d_{ap} + \alpha$).

From the above definitions, it can be clearly observed that all the triplets depend on the negative position; therefore, these three categories can be also called hard negatives, easy negatives and semi-hard negatives. Mathematically, hard positive, hard negative and semi-hard negative can be expressed as given in Equation 5.10, Equation 5.11 and Equation 5.12, respectively.

$$\underset{\boldsymbol{x}_i^p}{\operatorname{argmax}} \left|\left| f(\boldsymbol{x}_i^a) - f(\boldsymbol{x}_i^p) \right|\right|_2^2 \tag{5.11}$$

$$\underset{\boldsymbol{x}_i^n}{\operatorname{argmax}} \left|\left| f(\boldsymbol{x}_i^a) - f(\boldsymbol{x}_i^n) \right|\right|_2^2 \tag{5.12}$$

$$d_{ap} < d_{an}, d_{ap} - d_{an} + \alpha > 0 \tag{5.13}$$

In terms of mining of these triplets, there are two commonly used methods: offline mining and online mining. To sample triplets that violate the constraints imposed by the loss function, we can use two different methods to find the hardest candidates over the whole dataset (Offline triplet mining) or finding the hardest samples within the batch in the current iteration of the training (Online triplet mining). For now, we were unable to implement the online mining due to limited resources and due to the required large mini-batch size, which consumes lots of GPU memory. Online mining will be part of our future work. In this research, we implemented the offline mining approach.

5.8. Offline Mining Triplet

Offline triplet mining aims to sample triplets that break the assumptions of the loss function by the largest margin [77]. It means that it samples the positive image whose distance to the anchor is the largest and the negative image whose distance to the anchor is the smallest.

Finding only the hardest samples to train the model might not be a good idea because they can be over-complicated samples which can make the training diverge. The image on the left shows how the offline triplet mining is performed. The half-transparent green and red circles represent the positive and negative samples, respectively. The blue circle is the anchor. Finally, the dark circles represent the hard positive and negative sample chosen with respect to the anchor using this mining method.

In offline mining, triplets are found offline before the start of each training epoch. All the embedding are computed over the training dataset and then hard or semi-hard triplets are selected. Then, each epoch is trained over these selected triplets. We can then train one epoch on these triplets. Concretely, we would produce a list of triplets (i, j, k) and then create batches of these triplets of size B. This means that we will have to compute 3B embedding to get the B triplets and the loss of these B triplets and then back-propagate into the network. We need to do a full pass on the training set to generate triplets. An update of the offline mined triplets regularly is also required.

The possible combination of triplets in a dataset of M elements is $\binom{M}{3} = \frac{M!}{3!(M-3)!}$. By choosing a value of $M = 10000$, we already have 1.6×10^{11} possible combinations of triplets, thus making it impossible to sample all of them. To avoid the computation cost of sampling all the triplets, we could use random sampling, but this can generate triplets that do not break the constraints of the loss function. Thus, they do not contribute to the training of the model. By randomly sampling triplets, the converge of the training becomes slow, and given that the triplets used are not meaningful, the model could converge in a poor local minima. Because of this, it is crucial to have a powerful sampling technique that not only yields meaningful triplets to the training but also speeds up model convergence.

5.8.1. Triplet Sampling

The selection of triplet from a large number randomly is inefficient for training the network, and is slow. This makes triplet selection and the important aspect of the triplet loss function implementation. Hard triplets are those which actively participates in the training, and mining of hard triplets is a challenging task. In general, two main approaches used for mining of hard triplet are online and offline [143]. While implementing the triplet loss function, researchers have observed that choosing hardest pairs of triplets for the CNN can result in the local minima problem and in poor performance. The first technique involves the selection of negative samples randomly. Given an anchor vector for a given image, this technique involves taking a negative example randomly. This is the fastest method, but it does not guarantee that the model will learn to produce high-quality discriminative embedding.

Semi-hard sampling is another approach for the selection of triplets which at the first stage involves the computation of Euclidean distance between all anchors and positive samples in the same class. In the next step each negative samples distance from each positive sample is calculated. The selection of positive and negative vectors is performed by selecting the negative sample such that the distance between the anchor and negative is less than the distance between the anchor and its negative. This method results in a more effective way to force the model into focusing and bringing together anchors and positive samples before increasing the distance between anchor and negative samples.

The final method is called “hard sampling” or “hard sampling mining”. It requires the calculation of distances between all pairs of samples of the dataset. Then, the positive sample is drawn randomly. As the negative sample, we take the vector embedding with the closest distance from the current anchor vector. This constitutes a raw method, which leads to a difficult optimization problem. Starting with this method often causes the network to be stuck into a bad local minimum, thus producing a poor performance. This research uses an offline sampling methodology for triplet selection, which allows the CNN to adapt to triplet loss without arriving at the local minima.

In the future, more sampling methods should be explored. One of such method is online sampling. Using online sampling, hard negatives are chosen directly from the mini-batch. For this reason, a large batch size may be needed to make online methods work properly. Some research efforts have also be carried out to improve the stability of training through an online method which incorporates the loss from all triplets in a batch simultaneously rather than just the ones that are sent separately.

Summary

This chapter has detailed the various CNN architectures including AlexNet, RASNet, NASNet and MobileNet, that have been used in the experiments with Kaggle dataset. Furthermore, theoretical details about distance metric learning, triplet loss function, triplet loss training and triplet mining that has been proposed to enhance the performance of these CNN architectures are presented.

CHAPTER 6

EXPERIMENTS WITH DEEP LEARNING APPROACHES

6.1. Introduction

The performance of deep learning approaches in classifying distracted behaviours of drivers has been evaluated by conducting a series of experiments using the Kaggle distracted driving challenge dataset [2], where different configurations of deep architectures such as AlexNet, ResNet, MobileNet, and NASNet have been implemented and trained with various loss functions. In total, three sets of experiments have been carried out.

Initially, preliminary experiments were carried out using the AlexNet deep architecture, where the performance of the AlexNet model trained with cross-entropy loss function was used as the baseline performance to facilitate a comparison against the performance of the same AlexNet architecture trained with triplet and batch triplet loss functions, respectively. The aim of our preliminary experiments was to implement the simplest deep architecture on the Kaggle dataset and to explore whether the triplet loss and the batch triplet loss implementation would improve the accuracy of the trained model. From the results of our preliminary experiments, improvements in the classification accuracy were observed for both the triplet loss and the batch triplet loss functions.

Secondly, more complex and state-of-the-art deep models were implemented. The set of experiment 2 was designed to implement four particular deep architectures, AlexNet, ResNet, MobileNet and NASNet, with Softmax and SVC. The aim was to explore whether the SVC classification instead of Softmax improves the overall classification accuracies. The experimental results indicate that the SVC has slight advantage over the conventional Softmax classification.

Finally, in the third set of experiments, all four aforementioned deep models were implemented with the naïve triplet loss function, margin triplet loss function and batch triplet loss function. The aim of experiment 3 was to explore and identify the best deep

model among the three loss function configurations in terms of performance. Experiment 3 results also showed that the batch triplet loss function resulted in the best performance for all four models.

All the models in the experiments were trained over the Kaggle training dataset and were validated over the Kaggle test dataset. Results were submitted to Kaggle in the form of an Excel (.csv) sheet, and log loss public and private scores are used as a measure of performance.

Overall, this chapter examines how different experiments were designed in terms of different parameters selection, including details about the Kaggle dataset, libraries used for implementing the models and the experimental results.

6.2. Implementation

6.2.1. Preliminary Experiments

Three different models used in this set of experiments are listed as follows

- Modified AlexNet (Model A)
- Modified AlexNet with Triplet Loss Function (Model B)
- Modified AlexNet with Batch Triplet Loss Function (Model C)

The overall training dataset provided by Kaggle was used for both training and validation purposes in the preliminary experiments; while the dataset has been split into two subsets in the following two sets of experiments to support validation of trained models on unseen images.

The AlexNet model was implemented using the Caffe framework with an input image size of $(3 \times 227 \times 227)$ to the network. In all implemented models, Gaussian filters were used as weights in the convolutional layers with the stride of 4. What's more, the max pooling approach was used in the pooling layers and ReLU activation function for the feature activations. At the fully connected layer (fc7), 4096 feature representations were achieved, and the Softmax classifier was used to predict the output classes. In the model trained with the margin triplet loss function, the Softmax layer was replaced with the triplet loss layer,

and a margin α of 0.2 was used. Furthermore, this model implemented the L2 normalization approach and 50% dropout in order to avoid the overfitting of the model during the training process. A batch size of 128 has been used in the training, and the output classes number was 10 as defined by the Kaggle Challenge. Also, rather than sparse connectivity, dense connectivity was used in the convolutional layers because of its limited computational resources. Results were obtained from the validation dataset in the form of confusion matrices, accuracy curves, loss curves and roc curves. Furthermore, the results of Model A have been submitted to Kaggle to receive the log loss score.

6.2.2. Experiment 2: Softmax vs SVC

The following model configurations were tested in this experimental setup:

- AlexNet with Softmax (AlexNet+Softmax)
- AlexNet with SVC (AlexNet+SVC)
- ResNet50 with Softmax (ResNet50+Softmax)
- ResNet50 with SVC (ResNet50+SVC)
- MobileNet with Softmax (MobileNet+Softmax)
- MobileNet with SVC (MobileNet+SVC)
- NASNet with Softmax (NASNet+Softmax)
- NASNet with SVC (NASNet+SVC)

All the models were implemented using the Keras implementation framework with TensorFlow working on the backend. Apart from AlexNet, all the other models were imported from the Keras library and were used with an average pooling option. All the four models were already pre-trained on the ImageNet dataset. The pre-trained weights of AlexNet were obtained from [147]. The dimensions of the input images are $(3 \times 227 \times 227)$, $(3 \times 224 \times 224)$, $(3 \times 224 \times 224)$ and $(3 \times 331 \times 331)$ for AlexNet, RestNet, MobileNet and NASNet respectively. Also used were the implementations with the batch size of 64 with exception in the case of AlexNet where the batch size was 32. The training dataset was divided into 70% and 30% for training (22424 images in all the training dataset) and validation purposes. The test dataset (79726 images) is provided by the Kaggle and is unlabelled. To avoid overfitting, the dropout approach was used with the value of

0.4 for all the models except AlexNet where a dropout of 0.2 was used. A ReLU activation function with the batch normalization approach was used for the models in this set of experiment. Adam optimizer with a learning rate of 0.001and a decay of 0.0 was used with the cross-entropy loss while training the models with the Kaggle data. For all the models in this experiment set, the early stopping approach was used with the patience of 4 and "min" mode using the ModelCheckPoint. The feature representation of 4096, 2048, 1024 and 4032 with trainable parameters of about 4 million, 2 million and 3 million were achieved at the fully connected layer in AlexNet, ResNet50, MobileNet and NASNet models with Softmax classification, respectively. All the models were trained over 15 training epochs. Finally, the models were evaluated to validate and test the dataset. Results of the evaluation over validation dataset were saved in the form of accuracy/loss plots and confusion matrices. At the same time, results of evaluation for test dataset were saved as CSV files which were submitted to Kaggle, and log loss scores were received.

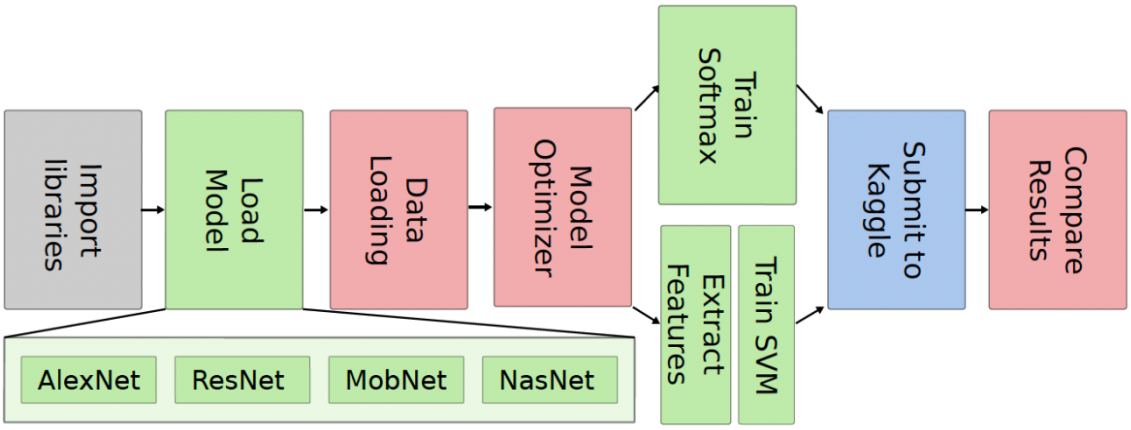


Figure 6.1: Summary of All the Processes Followed in Experiment 2.

First it was tried to adapt the pretrained models to the given distracted driver dataset using Softmax and negative log-likelihood. It was found out that this did not work properly. Therefore, tried to use the trained models as feature extractors and train a SVC with those features obtaining better results than with Softmax.

First experiment was carried out to see how a deep net model could perform in the distracted driver dataset using Softmax and negative log likelihood for classification. Figure 6.1 shows a diagram that summarizes this first experiment. For all other models except

AlexNet, 128 features were extracted at the fully connected layers with ReLU activation and were classified into 10 classes with Softmax activation giving us the probabilities of the input image to belong to each of the 10 classes of the distracted driver dataset. This was done in other to test the performance of the new trained models on the Kaggle dataset. All the deep net models have been pretrained on ImageNet before used as fixed feature extractor for the new Kaggle dataset, Once the features for all the images have been extracted, linear classifier was trained, in this experiment, Linear SVM and Softmax classifier for the new Kaggle dataset.

The Figure 6.2 shows the different architectures used for Softmax classification and for SVC of the feature vectors extracted by the net. It can be seen that how the last Softmax and fully connected layer are removed when the network is used as a feature extractor.

To make the network a feature extractor we removed the last two fully-connected layers of the model and changed it for a fully connected layer that has 128 features. Now the output of the network is the feature vector instead of the probabilities. This output feature vector was used as the input to a SVC model and trained the SVC with the hinge loss. Similar to Softmax classification, the SVC produces almost perfect confusion matrices with really low error in each class of the dataset.

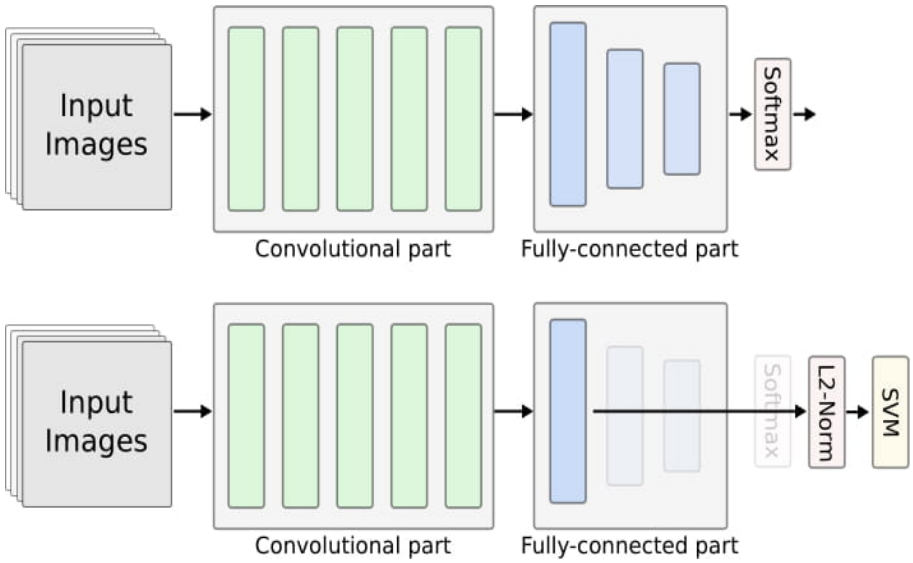


Figure 6.2: Comparison of the Architectures used for Traditional Softmax and SVC.

Classification

The trained CNN learns to map each input provided to a normalized embedding vector of 128 dimensions. It is worth mentioning that the embedding often needs to be normalized to have unit length, i.e., $\| \boldsymbol{x} \| = 1$, in order to be robust to illuminate and contrast changes and for training stability [148]. The spatial position, which each class ends up in, is not directly selected by the loss function as it is the case with sigmoid loss function training. For this reason, another model must be trained on top of the trained CNN to learn how to get class information from the embedding. To do this, the trained CNN is run to get embedding, and an SVC model is trained on the embedding. The reduction in input vector dimensionality has resulted in order-of-magnitude improvements in training convergence time compared to training the SVC model directly on the feature vector of a CNN. In this experiment two SVC kernels are tested: linear and radial bias function. In initial experiments, it was evident that the linear function outperforms radial bias kernel in this situation. Therefore, only linear kernel SVC model results are included in this report.

In the experimental result section, it will be investigated that how the deep net model using Softmax classification and negative log-likelihood for training gets worst accuracy than using the deep net model as a feature extractor and classifying those features with a SVC. We argue this happens due to overfitting problems where the SVC has better generalization capacity than the Softmax function.

6.2.3. Experiment 3: Triplet Loss

If the classes in the feature space learned by a CNN are well separated, it would be even easier for a SVC model to learn the parameters required to distinguish between different classes. Triplet loss refers to a loss function which is used to direct a CNN to learn such a feature space. The model configurations tested in this experimental setup are as follows:

- AlexNet with Margin Triplet Loss (AlexNet+Margin)
- AlexNet with Naïve Triple Loss (AlexNet+Naïve)
- AlexNet with Batch Triplet Loss (AlexNet+Batch)
- ResNet50 with Margin Triplet Loss (ResNet50+Margin)
- ResNet50 with Naïve Triple Loss (ResNet50+Naïve)

- ResNet50 with Batch Triplet Loss (ResNet50+Batch)
- MobileNet with Margin Triplet Loss (MobileNet+Margin)
- MobileNet with Naïve Triple Loss (MobileNet+Naïve)
- MobileNet with Batch Triplet Loss (MobileNet+Batch)
- NASNet with Margin Triplet Loss (NASNet+Margin)
- NASNet with Naïve Triple Loss (NASNet+Naïve)
- NASNet with Batch Triplet Loss (NASNet+Batch)

Like in set of experiment 2, in this set of experiments all models were implemented using Keras framework with Tensorflow at the backend. Input image dimensions were the same for each model as in experiment 2 with same batch sizes. Furthermore, dataset division was identical to experiment 2: 70% of training dataset for training and 30% of training dataset for the validation. Certain pre-processing was performed on the image before it was subjected to the model. For triplet selection, a semi-hard sampling mode was used. All triplet loss is calculated with a margin $\alpha = 0.3$ and $\beta = 0.5$ for the batch triplet loss. A dropout of 0.4 was used in all the models to avoid overfitting during the training process. Adam optimizer with a learning rate of 0.0001and decay of 0.005 was used along with naïve, margin and batch triplet loss functions for different deep models. Finally, based on the experiment 2 results which indicated that SVC is a better classifier than Softmax, for this experiment, SVC was used at the fully connected layer in place of Softmax for the final classification. Overall, 32 training epochs were used for training, and models were evaluated with the validation dataset and test dataset. On validation dataset results in the form of accuracy and loss were achieved, while on test dataset, results in the form of Kaggle loss score were achieved.

Triplet Sampling Implementation

Sampling begins with the selection of 600 samples of images from the various classes of safe and distracted driving, with taking sampling probabilities from each class all coming from the Kaggle dataset. These are taken randomly, two samples are added at a time, where anchor and positive are of the same category predictions made in sets of 3 for anchor, positive and negative. After this is complete, one of three methods (random, semi-hard or

hard) is applied to get negative samples. After pre-processing, triplets sampling is performed with semi-hard negatives using an offline approach. A negative sample is selected at random with forward pass choosing either hard or semi-hard negative samples, this is done in other to generate triplets. A data generator for triplets with a batch-size of 3×100 for the size of a mini-batch is selected. In the next step, triplet loss functions were defined, and standard squared Euclidean distances were calculated all over the batch for standard triplet loss considering the distance from the anchor and positive d_{ap} and for the anchor and negative, d_{an}, means over the batch, μ_{ap} and μ_{an} for anchor and positive and anchor and negative respectively for naive and batch triplet loss as well as average variance for batch triplet loss. The model was built with the model.fit() function provided by Keras using the Adam algorithm for optimization specifying a margin triplet loss as a loss function and setting a learning rate of 0.0001 and a decay rate of 0.0005, as well as a L2 norm for regularization.

For the experiments using the triplet losses with AlexNet, we have a total of 40611328 trainable parameters, zero non-trainable parameters and an input size of $3 \times 227 \times 227$. For the experiments with ResNet50, we used an input shape of $(224 \times 244 \times 3)$ for 23849984 total trainable parameters and 53120 non-trainable parameters. We also utilised an input shape of $(331 \times 331 \times 3)$ for NASNet with 85433042 total parameters from which 85236374 are trainable and 196668 non-trainable parameters. For MobileNet, the input shape is $224 \times 224 \times 3$, 3360064 total parameters, from where 3338176 are trainable and 21888 are non-trainable parameters.

In this set of experiments, an early stopping was performed for each model using a validation dataset made of 30% of the total training set to define that we wanted to monitor the validation loss at each epoch and after the validation loss has not improved for 10 consecutive epochs, then training is interrupted.

Training Method

All four models used in this experiment set are pre-trained on the ImageNet dataset. Then, each of the models is trained on 96 epochs of 300 triplets, where one triplet consists of three images. This number of triplets is selected to provide a good balance between being able

to gather effective hard and semi-hard negatives and being able to adjust to what the model learns during training. We denote completion of training for one of these sets of 300 images as an epoch. Summarizing:

- Each Epoch consists of 30 computations
- 1 computation consist of 10 triplets
- 30 computation consist of $30 \times 10 = 300$ triplets
- 1 triplet consists of 3 images
- 300 triplet consist of 900 images.

For all models except AlexNet, the first 32 epochs of training use random negatives triplets, the next 32 epochs use semi-hard negatives, and the last 32 utilise hard negatives. On AlexNet, this training regimen led to poor results, as the model never adapted to the point where it was successful at optimizing triplet loss on semi-hard or hard samples. The only moderately successful routine for training AlexNet with triplet loss was pure random sampling. This could be due to the limited capacity and the depth of the AlexNet model relative to other architectures used.

The NASNet model implemented in this experiment is the NASNetLarge, and that is why NASNet is trained with the same batch size. However, the training is distributed over a 6GPU HPC cluster in a data-parallel fashion. This was necessary for training NASNet with a batch size of 12, considering the larger input size of (331, 331, 3) to the model. NASNet used the same training process as MobileNet and ResNet50.

All models were trained using all three implementations of the triplet loss function described above. The models are trained using the Adam algorithm for first-order gradient-based optimization with a learning rate of 0.0001 and a decay rate of 0.0005. Again, all models except AlexNet used 50% dropout to ensure training robustness while AlexNet used 40% dropout. Approximately 30 % of all available training data is withheld from the training process and used to validate the classifier. A batch-size of 30 is used. The number used must be a multiple of three for the triplet loss calculation.

Figure 6.3 has a summary on how to perform the training using triplet loss functions. We trained the four convolutional architectures using three different triplet loss functions: batch

triplet loss, margin triplet loss and naive triplet loss. Once trained we use the models as feature extractors that map the input image into an Euclidean feature space where we can compute distances. Then, we input those feature vectors into a SVC trained using two different kernels: linear kernel and a radial basis function. At the end we compare the results and we found out the linear kernel consistently outperforms the radial basis function one. The final predictions are submitted to the distracted driver Kaggle competition obtaining really low error rates.

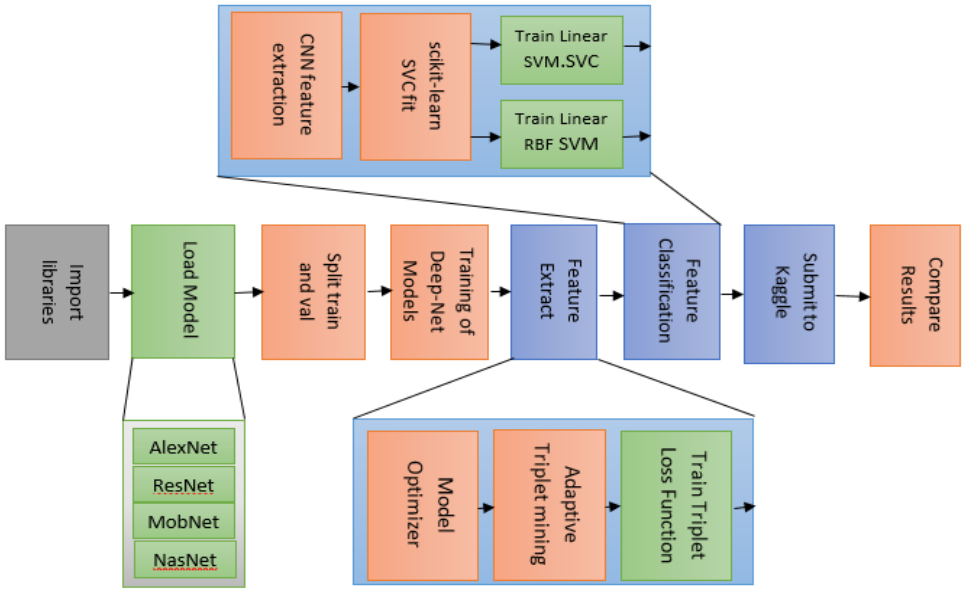


Figure 6.3: Summary of Process Followed in Experiment 3.

In Figure 6.4, first we forward the input images of the whole dataset to the network, then we are capable to compare distances between the obtained feature vectors and sample the triplets according to the strategy that corresponds regarding the training iteration. The triplet sampling is performed obtaining semi-hard negative samples or hard negative samples. After obtained the triplet we compare the distances between the anchor and positive image and the anchor and negative image and add a margin to finally obtain the value of the triplet loss function (the error). We optimize on this error applying then backpropagation to the CNN in order to refine the Euclidean space where the generated feature vectors lie. After training, the feature vectors that belong to images of the same

semantic class are placed together in the Euclidean space while the feature vectors of the images that have different semantic class are placed with a large distance between them.

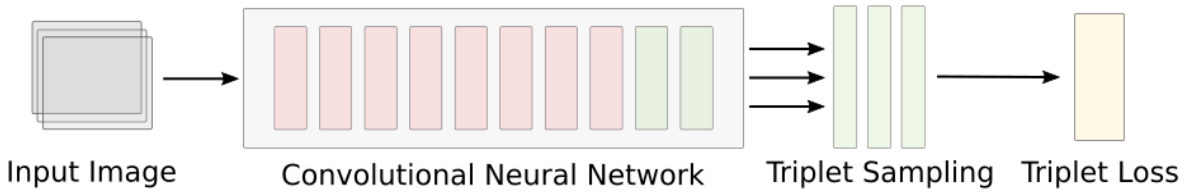


Figure 6.4: Example of Single Training Instance using Adaptive Triplet Sampling Strategy.

Feature vector classification

After training, the network can produce high-quality feature vectors capable of encoding all the image information. The feature vectors are placed in an Euclidean space together with other samples of the same semantic class while far away of items with different label, but, the network is only capable of producing a feature vector with 128 features. Instead we want the probabilities of the input image $\boldsymbol{x}_i$ to belong to each of the 10 classes of the distracted driver dataset. Therefore, the feature vectors are classified using a SVC that will give us the probabilities of the input image $\boldsymbol{x}_i$ to belong to each of the 10 classes. Using these feature vectors, we can compute the Euclidean distances and directly apply the loss functions that we described to place the input images $\boldsymbol{x}_i$ into a feature space where clustering by distance can be applied. Due to the characteristics of the dataset, where similarity between the different actions plays a key role, a training by similarity is highly suitable.

Figure 6.5 shows the pipeline of the classification of the features extracted from the neural network. The CNN has been pretrained in the ImageNet dataset and then fine-tuned using Kaggle distracted driver dataset. This allows it to extract high-quality feature vectors that are then plugged into the SVC (in this case with a linear kernel) that is trained using the hinge loss in order to perform an accurate classification of the input image into the ten classes of the distracted driver dataset.

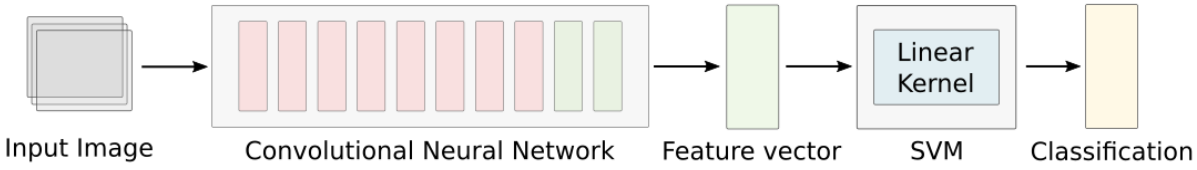


Figure 6.5: Pipeline for the Classification of Features Extracted from Neural Network.

All models were trained using the Adam optimizer [85], where we set empirically set a learning rate of 0.0001 with a weight decay of 0.0005 and standard β values of 0.9 and 0.999 respectively. To ensure generalization and robustness during training, we set the dropout [87] rate in 0.4 except in AlexNet where we found out that Dropout made the training diverge instead of helping it to converge.

To avoid high feature values and give robustness to the SVC training, we normalize the feature vectors to have unit norm $||M(\boldsymbol{x}_i)|| = 1$ before feeding it to the SVC. We tested two different kernels that are applied before performing the classification, linear kernel with formula $k(\boldsymbol{x}_i, \boldsymbol{y}_i) = \boldsymbol{x}_i{}^T \boldsymbol{y}_i + c$ and RBF kernel, with formula $k(\boldsymbol{x}_i, \boldsymbol{y}_i) = e^{-\frac{\|\boldsymbol{x}_i - \boldsymbol{y}_i\|^2}{2\sigma^2}}$ where σ is a trainable parameter.

The SVC with the linear kernel consistently outperforms the RBF kernel. Therefore we only show the results obtained using the linear kernel for the SVC and getting the input features with the margin, naïve and batch triplet loss training in the CNN.

6.3. Kaggle Dataset

Dataset provided by the Kaggle challenge for distracted driving classification [2] has been used in this research for the investigation on deep learning approaches. Kaggle training dataset includes 22,424 two dimensional RGB images of 480 × 680 size divided into 10 different classes. One of the 10 classes is for safe driving, and the other 9 classes include the images of drivers involved in different distraction related activities such as eating, talking on phone, texting, makeup, reaching behind, adjusting radio, or conversing with other passengers. All images in the dataset have been captured using a single vehicle dashboard optical sensor (camera) [2]. Table 6.1 presents the details of all the 10 classes

defined by the Kaggle challenge and the exact number of images in each class. Figure 6.6 presents example images from each of the Kaggle challenge classes. In all experiments, the Kaggle images were pre-processed before they were subjected to deep learning models and the images were processed for resizing and normalizing based on the means obtained from the initial training of deep learning models.

Table 6.1: Prediction Classes for Kaggle Task and Number of Images in Each Class [2].

Class0	**Class1**	**Class2**	**Class3**	**Class4**	**Class5**	**Class6**	**Class7**	**Class8**	**Class9**
Safe Driving	Texting-Right	Talking on Phone-Right	Texting-Left	Talking on Phone-Left	Operating the Radio	Drinking	Reaching Behind	Hair and Makeup	Talking to Passenger
2489	2267	2317	2346	2326	2312	2325	2002	1911	2129

Class 0 Class 1 Class 2 Class 3 Class 4

Class 5 Class 6 Class 7 Class 8 Class 9

Figure 6.6: Sample Images from Each Kaggle Driver Distraction Challenge Class.

6.4. Implementation Frameworks

In the literature, researchers have used a number of implementation frameworks to implement deep learning algorithms such as Caffe, Keras with TensorFlow, Torch and PyTorch. All the mentioned implementation platforms have their own advantages and disadvantages. Table 6.2 presents the comparison of the most commonly used deep learning frameworks in the literature.

For the preliminary experiments of this research, Caffe [149] was used as the implementation framework because it was easy-to-use, the best choice for beginners and the most popular choice at the time of the experiments. Caffe is a deep learning architecture created by Yangqing Jia at Berkeley AI Research (BAIR) during his PhD from UC Berkeley. This framework was developed to facilitate the researchers by providing expressive structure, faster speeds and modularity. This library contains a number of pre-trained deep architectures such as AlexNet, which has been used in this research.

Table 6.2: Comparison of Common Deep Learning Implementation Frameworks.

Framework	License	Open Source	Core Language	Interface	Pre-trained Models	CUDA Support
Caffe [149]	BSD	Yes	C++	Python, C++, MATLAB	Yes	Yes
Keras [150]	MIT	Yes	Python	Python, R	Yes	Yes
MATLAB [151]	Proprietary	No	C, C++, MATLAB	MATLAB	Yes	Yes
PyTorch [152]	BSD	Yes	Python	Python	Yes	Yes
TensorFlow [153]	Apache	Yes	C++, Python	Python, C++, Java, R	Yes	Yes
Theano [154]	BSD	Yes	Python	Python	Yes	Yes
Torch [155]	BSD	Yes	C, Lua	C, C++	Yes	Yes

For experiment sets 2 and 3, Keras [150] with TensorFlow [153] backend was used as the deep learning framework API. All the scripts were written in Python 3 programming language and supported by a number of libraries such as Theano, Scipy, Numpy, and Pandas, and part of SciPy and Sklearn ecosystems were used. Together with Keras, we have used the well-known scikit-learn package in order to train the SVCs for classification. Similar to Keras, scickit-learn is a high-level API that allows the users to train machine learning models easily and abstracting the user from the mathematical and low-level formulation that it requires. To create learning curves and confusion matrices we have used the Matplotlib library which easily allows the users to create line plots or graphs. Also, Jupyter Notebooks have played an important role. Jupyter Notebook is similar to a development environment that puts interactivity as a priority, making it an essential tool

for machine learning projects to show graphs, confusion matrices and other kinds of visualization.

6.5. Evaluation Criteria

Well-defined evaluation criteria are significant in assessing experimental performances. In this research, the performance of the trained deep learning models is assessed based on the classification accuracy over validation data and evaluated with the Kaggle log loss scores obtained from the Kaggle submissions. Overall, the training dataset provided by the Kaggle is divided into certain ratio for training and validation. For the preliminary experiments, the Kaggle training set has been divided in a ratio of 75% to 25% for training and validating the proposed algorithms, respectively. For experiment sets 2 and 3, the split was 70% and 30%. Models are trained over the training set and validated for their performance on the validation dataset in terms of classification accuracies and the loss. Then, algorithms are subjected to the test dataset provided by the Kaggle. Because the correct labels of the test dataset provided are not public and only Kaggle has the access to those labels, results over the test dataset are submitted to Kaggle in the form of CSV file, and public and private log loss scores are achieved. Given that it is a loss, smaller values of Kaggle loss score indicates better performance. Finally, in terms of real-world implementation, models in experiment 2 and experiment 3 are compared in terms of their sizes and processing times.

6.5.1. Kaggle Scores

A total of 22424 images was provided by Kaggle to the Kagglers for the purpose of training their algorithms, and they were asked to submit the classification results for 79,726 test images in the form of an Excel sheet in csv format. Kaggle evaluated each submission and issues a loss score using a multiclass log loss function given in Equation 6.1.

$$\text{logloss} = \frac{1}{N}\sum_{i}^{N}\sum_{j}^{M} y_{ij}\log(p_{ij}) \tag{6.1}$$

where N denotes the total number of images, M denotes the total number of classes, y_{ij} denotes the actual class of image and p_{ij} denotes the predicted class of the image.

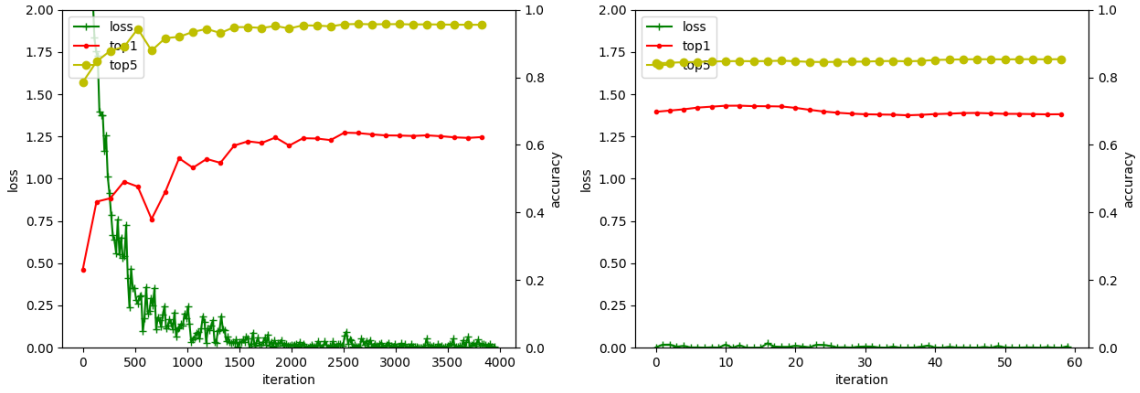
6.6. Experimental Results

This section presents the results and discussions on the three sets of experiments performed using the deep learning approaches. Results are critically analysed, and important insights have been reported.

6.6.1. Preliminary Experiment Results

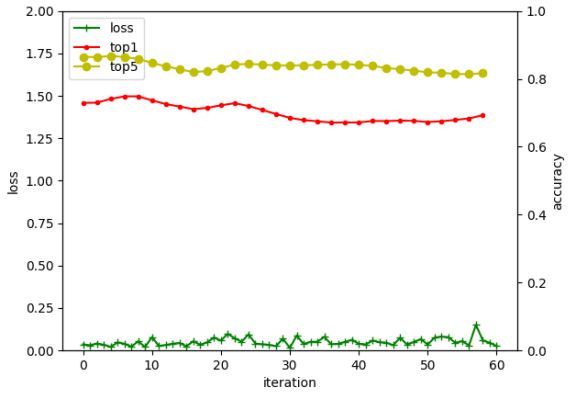
AlexNet based deep architectures with Softmax, triplet loss and batch triplet loss were tested against the Kaggle dataset to evaluate their performance. Results are presented in the form of confusion matrices and accuracy/loss plots. Model A (AlexNet+Softmax) was trained over 3960 iterations, while Model B (AlexNet+Triplet Loss) and Model C (AlexNet+Batch Triplet Loss) were further trained over 60 iterations. The purpose of the preliminary experiments was to have insights into the fundamentals of deep learning implementation, including the effectiveness of triplet loss in improving the classification accuracy.

For the accuracy assessment, top 1 and top 5 accuracies have been determined for all three models. Top 5 accuracy is also a credible measure to assess the working performance of algorithms in computer vision. The idea behind top 5 accuracy is that a prediction is considered as accurate if the correct class is included in the top 5 predictions produced by the algorithm. Hence, the top-5 accuracy is definitely a number with higher values than top 1 accuracy. Figure 6.7 present the accuracy and loss plots for all three models in the preliminary experiments, respectively. From figure, it can be observed that top 1 and top 5 accuracies improved as the number of training iterations increases, with top 5 accuracy always remain above the top 1 accuracy. Furthermore, the loss was reduced to nearly zero after 2000 training iterations. The behaviour of accuracy and training plots was normal in reference to the literature as in machine learning accuracy plots more or less follows a positive exponential curve, while loss plots follow a negative exponential curve. In Figure 6.7 (a) and Figure 6.7 (b), the accuracy and loss plots are more or less straight lines as the training was performed on a pre-trained model. From Figure 6.7, it can be observed that the training with all three deep learning approaches has converged.



(a) Model A

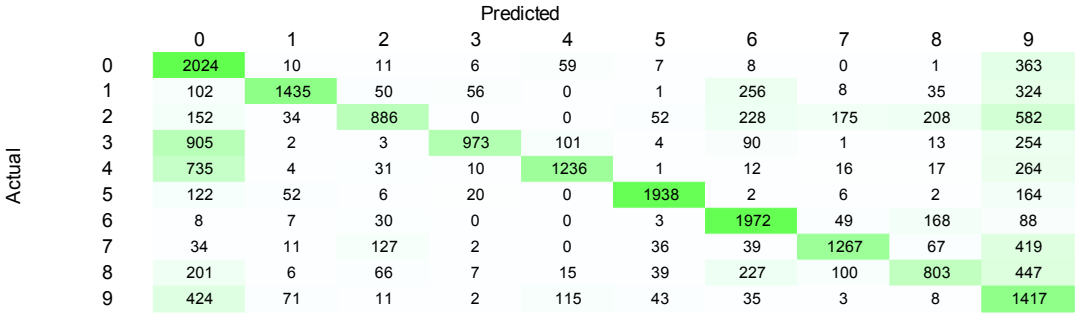
(b) Model B



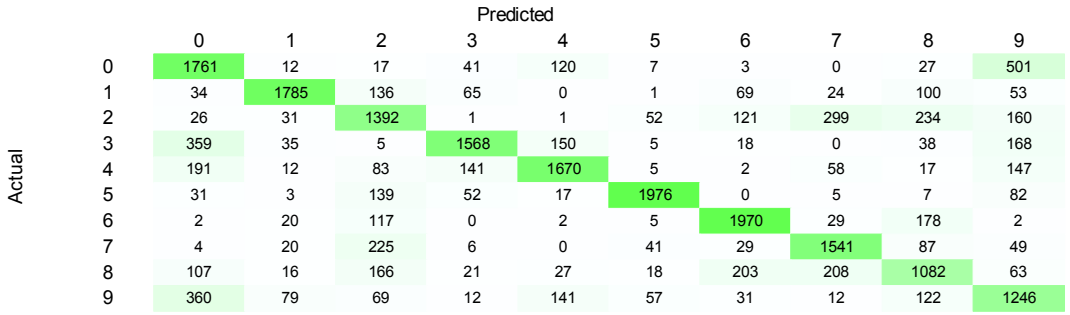
(c) Model C

Figure 6.7: Classification Accuracy and Loss Plots for all Models in Preliminary Experiment.

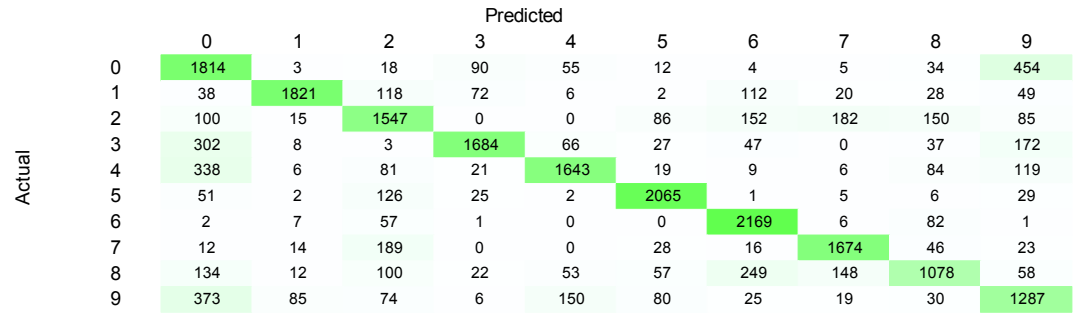
Proposed deep models were validated against the Kaggle dataset, and confusion matrices were plotted to visualize the results of their classification performance. Figure 6.8 show the confusion matrices for all three models ((a) Model A, (b) Model B and (c) Model C), respectively. From the confusion matrices, it can be observed that Model A classification results were lowest among the three, while Model C results were the best. Furthermore, while interpreting the results in more detail, it can be observed that there are couple of classes in which mis-classification percentages are high and prominent in comparison to other classes. Two such examples include class 7 misclassified as class 2 and class 9 misclassified as class 0.



(a) Model A



(b) Model B



(c) Model C

Figure 6.8: Confusion Matrices for All Three Models in Preliminary Experiments.

Figure 6.9 and Figure 6.10 shows some instances of correct classifications of class 7 and class 9 as well as wrong classifications of class 7 as class 2 and class 9 as class 0, respectively. From the figures, it can be suggested that the mis-classifications are due to similarity between two classes. In case of class 7 and class 2, only right hand of driver is different i.e. using mobile phone in class 2 while reaching back in class 7. Other than the

right hand, there are no other noticeable differences in images between the two classes since driver has same head posture, body orientation and left hand on steering. This situation makes it difficult for the algorithm to clearly differentiate and results in mis-classification. Similarly, in case of class 9 and class 0, there are a high number of mis-classifications which are due to a high similarity between the two classes. The only noticeable difference is the position of the head. Otherwise, both hands of the driver are on steering wheel and the driver has highly similar body position. This situation makes it highly probable for algorithm to mis-classify between the two classes as demonstrated by the results in confusion matrices.

(a) True Positives Class 7

(b) False Negatives Class 2

Figure 6.9: Instances of Correct Classification as Class 7 and Wrong Classification for Class 7 as Class 2.

(a) True Positives of Class 9

(b) False Negatives of Class 0

Figure 6.10: Instances of Correct Classification as Class 9 and Wrong Classification for Class 9 as Class 0.

Among the errors caused by mis-classification, the most dangerous error in our problem scenario of detection of distracted driving is to mis-classify a distracted driving as safe, i.e. in the case of Kaggle challenge mis-classifying an image from one of classes 1 to 9 as class 0 and vice versa. Table 6.3 shows the percentages of the images in each Kaggle class being recognised as safe driving and distracted driving for all three models, respectively. Table 6.4 presents the percentages of images of distracted driving (i.e. the images in Kaggle classes 1-9) being classified as its correct class or an incorrect class of distracted driving.

Table 6.3 shows that for Model A, 38.58%, 31.60%, 10.52% and 19.92% of images in class 3, class4, class 8 and class 9 have been mis-classified as safe driving, respectively. These miss-classifications are assumed to be associated with the similarities in images of some classes and indicate the failure of Model A to extract more meaningful features to support classification. These miss-classifications were reduced to some extent by Model B and Model C. However, there were still some high number of miss-classification percentages such as 15.30% for class 3 by Model B and 12.87% by Model C and 16.91% for the class 9 by Model B and 17.52% by Model C. As illustrated in Figure 6.10, a high level of similarity exists between class 0 and some images in class 9. Figure 6.11 illustrates the high level of similarity between class 0 and some images in class 3. The percentages of the other classes being mis-classified as safe driving have all been reduced to less than 10%. This indicates that use of triplet loss has improved the deep net performance by some extent.

Table 6.3: Percentages of Images being Classified as Safe and Distracted for All Three Models

Class Label		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
		True +ev	**False -ev**								
Classified as Safe Driving	Model A	81.32	4.50	6.56	38.58	31.60	5.28	0.34	1.70	10.52	19.92
	Model B	70.75	1.50	1.12	15.30	8.21	1.34	0.09	0.20	5.60	16.91
	Model C	72.88	1.68	4.32	12.87	14.53	2.21	0.09	0.60	7.01	17.52
		False +ev	**True -ev**								
Classified as Distracted Driving	Model A	18.68	95.50	93.44	61.42	68.40	94.72	99.66	98.30	89.48	80.08
	Model B	29.25	98.50	98.88	84.70	91.79	98.66	99.91	99.80	94.40	83.09
	Model C	27.12	98.32	95.68	87.13	85.47	97.79	99.91	99.40	92.99	82.48

Table 6.4 presents the results from another important aspect. It shows that when images are classified as distracted driving, how many percent of them has been classified to the correct distraction class and how many wrong. Table 6.4 shows that by Model A class 2 and class

8 both have a higher chance to be classified as a wrong distraction than the correct class of distraction. Models B and C have produced better classification results by replacing the Softmax of Model with a triplet loss function. However, for some classes such as classes 2, 8 and 9 the percentages of mis-classifications are still unsatisfactory. Such low classification performance is because the AlexNet used by all three models in our preliminary experiments was trained from scratch over the limited dataset of Kaggle rather than, like in our experiment sets 2 and 3, pre-train on some huge benchmark dataset like ImageNet and then further trained on Kaggle dataset.

(a) True Positive of Class 3

(b) False Negative of Class 0

Figure 6.11: Instances of Correct Classification as Class 3 and Wrong Classification for Class 3 as Class 0.

Table 6.4: Percentages of Images being Classified as Correct and Wrong Distracted Driving for All Three Models.

Class Label		1	2	3	4	5	6	7	8	9
Classified as Correct Distraction	Model A	63.30	38.24	41.47	53.14	83.82	84.82	63.29	42.02	66.56
	Model B	78.74	60.08	66.84	71.80	85.47	84.73	76.97	56.62	58.53
	Model C	80.36	66.77	71.78	70.64	89.32	93.29	83.62	56.41	60.45
Classified as Wrong Distraction	Model A	32.20	55.20	19.95	15.26	10.90	14.84	35.01	47.46	13.53
	Model B	19.76	38.80	17.86	19.99	13.19	15.18	22.83	37.78	24.57
	Model C	17.96	28.92	15.35	14.83	8.48	6.62	15.78	36.58	22.03

Table 6.5 presents the summary of experimental results for all three models. In the experiments, top-1 accuracy of 62.21%, 71.31% and 74.84% have been observed for Model A, Model B and Model C, respectively. It can be seen that Model C has the highest value of top-1 accuracy. Top-5 accuracy of 95.59%, 97.35% and 98.14% has been recorded for Model A, Model B and Model C, respectively. Again, it can be observed that Model C has shown the highest classification accuracy. To test the Kaggle dataset, results from the Model A were submitted to the Kaggle and a loss score of 1.55 and rank of 500+ out of 2000+ was achieved. Model B and Model C results were not submitted to Kaggle at that time; hence, they cannot be compared.

Table 6.5: Summary of Experimental Results of Preliminary Experiment.

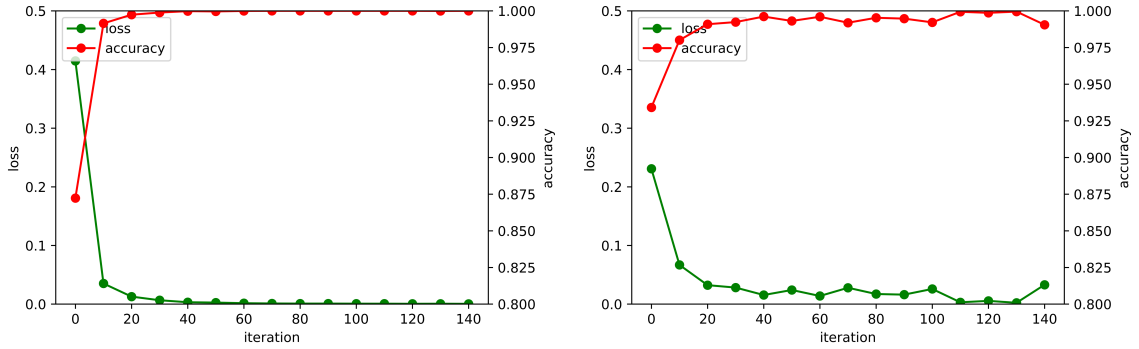
	Top 1 Accuracy	Top 5 Accuracy	Test Loss
AlexNet+Softmax	62.21	95.59	0.008658
AlexNet+Triplet Loss	71.31	97.35	0.004345
AlexNet+Batch Triplet Loss	74.84	98.14	0.009093

Satisfactory results were achieved from the preliminary experiments, thus indicating the potential of deep architectures in the distraction detection task. Through preliminary experiments, a basic understanding of the practical implementation of deep learning algorithms was established, and a familiarisation with different tools was achieved.

6.6.2. Experiment 2 Results

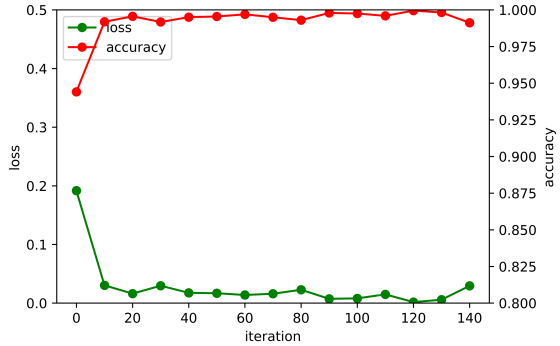
Based on the results of preliminary experiments which identified that deep architectures are potential candidates for being used in distraction detection effectively, more detailed and comprehensive experiments have been be performed using the state-of-the-art deep architectures rather than utilising only the basic AlexNet. In addition, experiment set 2 was performed to compare the performance of the Softmax classifier against SVC for four different state-of-the-art deep models. All models were modified to carry out the distraction classification task. It is worth mentioning that the AlexNet reported in this subsection is implemented and trained in Tensorflow and different from the one reported in the preliminary experiments.

First, the performance of all four models with Softmax during the training process was recorded. Figure 6.12 presents the accuracy and loss plots for all four models during the training process. Overall, the behaviour of models during the training was in accordance with standard training process i.e. the accuracy increases and the loss decreases with an increase in the number of training iterations. All the models were able to achieve high accuracies (over 98%) for the training dataset. The AlexNet model quickly converged towards a high accuracy, showing a sign of overfitting. However, other models such as ResNet and MobileNet indicated a slightly normal behaviour. The most appropriate training behaviour from all four models was shown by the NASNet architecture as it almost followed the exponential curves (positive exponential for accuracy plot and negative exponential for loss plot). Although the performance of models during the training are not considered to be much significant in terms of evaluating the overall functional performance of models because it only considers the data already known to the model, it is important to monitor these curves to ensure that models are well-trained.

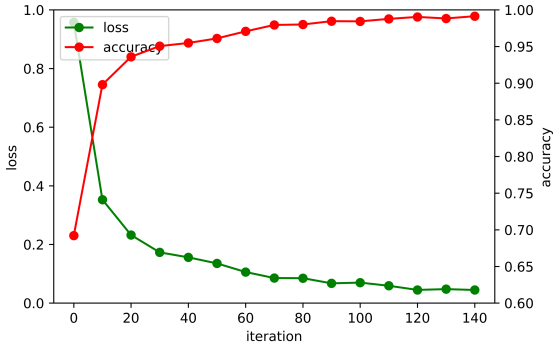


(a) AlexNet

(b) ResNet



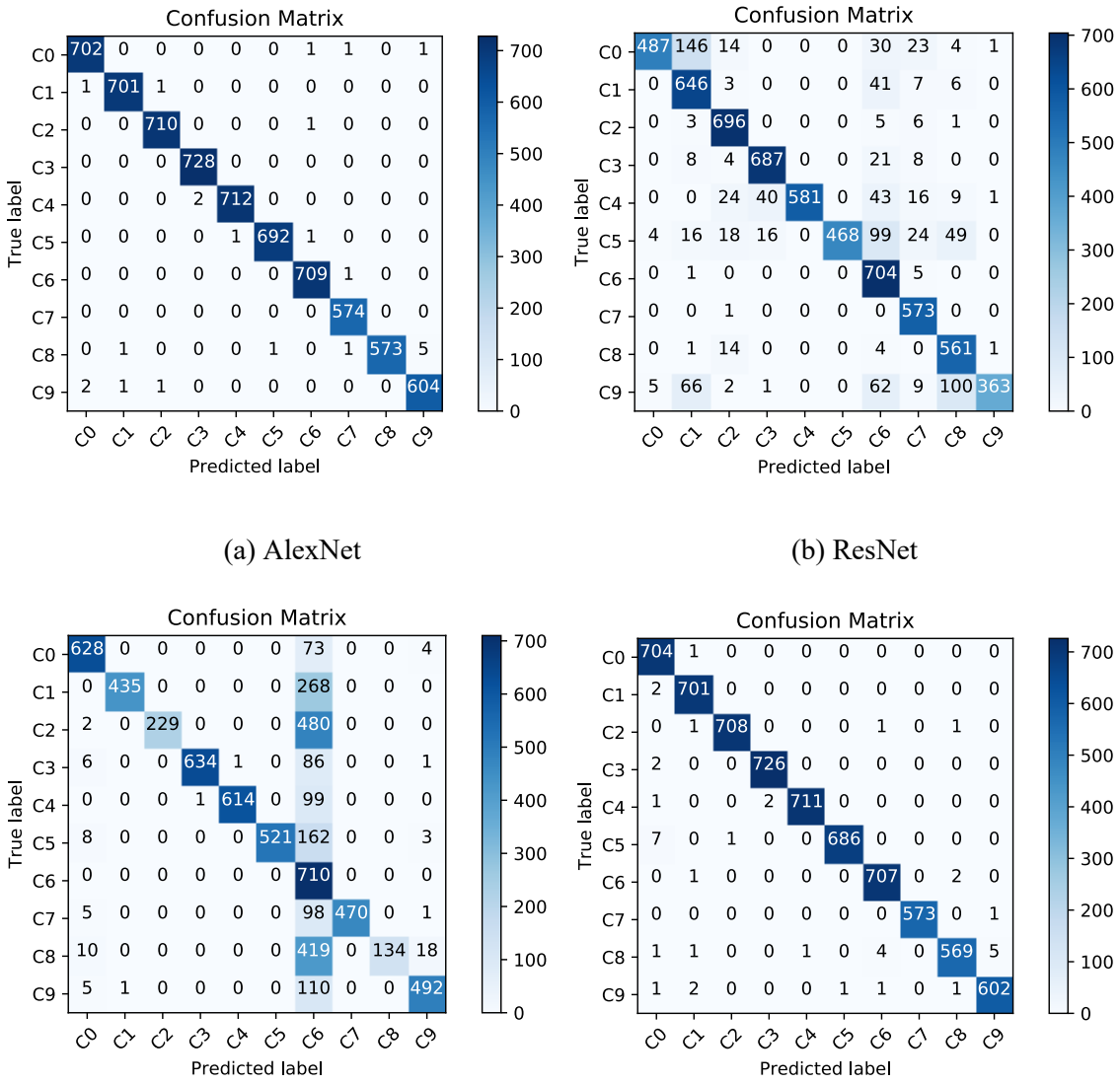
(c) MobileNet



(d) NASNet

Figure 6.12: Training Accuracy and Loss Plots for All Models with Softmax Classifier

Once the models were trained, they were validated over the validation dataset, and the results of their predictions were plotted as confusion matrices as shown in Figure 6.13 for Softmax and Figure 6.14 for SVC, respectively. Based on the confusion matrices, it can be observed that for the Softmax classifier, AlexNet and NASNet outperformed MobileNet and ResNet in terms of performance over the validation dataset. AlexNet even outperformed ResNet over validation dataset which was unusual compared to what has been reported in the literature. AlexNet accuracy was almost 100 percent with few wrong predictions, while NASNet accuracy was about 98 percent with some wrong predictions. On the other hand, ResNet and MobileNet performance over validation data was not as accurate. For the SVC, all four models performed almost the same with over 99% correct predictions and very few incorrect predictions.



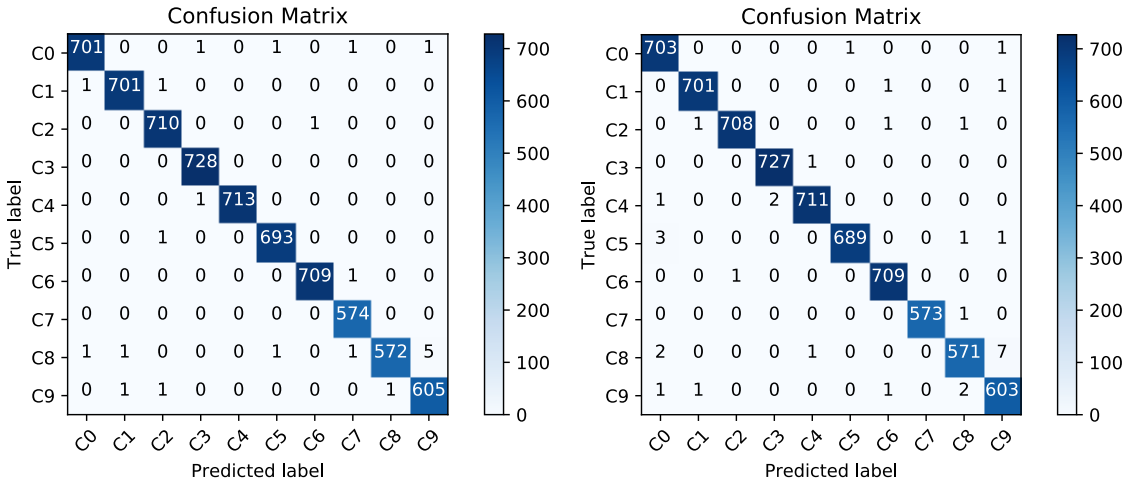
(a) AlexNet

(b) ResNet

(c) MobileNet

(d) NASNet

Figure 6.13: Confusion Matrices for Models with Softmax Classifier over Validation Dataset.



(a) AlexNet

(b) ResNet

(c) MobileNet

(d) NASNet

Figure 6.14: Confusion Matrices for Models with SVC over Validation Dataset.

Table 6.6 and Table 6.7 present the validation results of the proposed algorithms in experiment set 2. Table 6.6 presents the percentages of images in each dataset class being classified as safe or distracted by each of the models evaluated in this experiment set. Since it is very important for an algorithm not to classify a distracted driving as safe, these percentages are critical. From the Table 6.6, it can be observed that in all eight configurations of CNN models, there were very few miss-classifications from distracted driving classes into safe driving classes. Surprisingly, AlexNet results in Table 6.6 were

nearly perfect in comparison to the other state of the art models. However, from the training accuracy plot shown in Figure 6.12, the training of AlexNet converged rapidly and there was possibly overfitting during the training. Therefore, although AlexNet performance on the validation dataset is almost over 99%, its accuracy on the test dataset is significantly lower as shown in Table 6.8. Among other three models, MobileNet with SVC was the best in terms of least number of images being classified as safe driving from the distracted driving classes. Highest percentage of incorrect classification was observed as 0.49% for the case of class 9.

Table 6.6: Percentages of Images being Classified as Safe and Distracted for All CNN Model Configuration in Experiment 2.

Class Label		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
		True +ev	**False -ev**								
Classified as Safe Driving	AlexNet + Softmax	99.57	0.14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33
	AlexNet + SVC	99.43	0.14	0.00	0.00	0.00	0.00	0.00	0.00	0.17	0.00
	ResNet + Softmax	69.08	0.00	0.00	0.00	0.00	0.58	0.00	0.00	0.00	0.82
	ResNet + SVC	99.72	0.00	0.00	0.00	0.14	0.43	0.00	0.00	0.34	0.16
	MobileNet + Softmax	89.08	0.00	0.28	0.82	0.00	1.15	0.00	0.87	1.72	0.82
	MobileNet + SVC	99.57	0.28	0.00	0.00	0.00	0.29	0.00	0.00	0.00	0.49
	NasNet + Softmax	99.86	0.28	0.00	0.27	0.14	1.01	0.00	0.00	0.17	0.16
	NasNet + SVC	99.86	0.28	0.00	0.27	0.14	1.15	0.00	0.00	0.00	0.00
		False +ev	**True -ev**								
Classified as Distracted Driving	AlexNet + Softmax	0.43	99.86	100	100	100	100	100	100	100	99.67
	AlexNet + SVC	0.57	99.86	100	100	100	100	100	100	99.83	100
	ResNet + Softmax	30.92	100	100	100	100	99.42	100	100	100	99.18
	ResNet + SVC	0.28	100	100	100	99.86	99.57	100	100	99.66	99.84
	MobileNet + Softmax	10.92	100	99.72	99.18	100	98.85	100	99.13	98.28	99.18
	MobileNet + SVC	0.43	99.72	100	100	100	99.71	100	100	100	99.51
	NasNet + Softmax	0.14	99.72	100	99.73	99.86	98.99	100	100	99.83	99.84
	NasNet + SVC	0.14	99.72	100	99.73	99.86	98.85	100	100	100	100

Table 6.7 presents when images are classified as distracted driving, how many percent of them has been classified to the correct distraction class and how many wrong. From the

table, it can be observed that other than MobileNet with Softmax and ResNet with Softmax, all other configurations were able to classify the distraction classes nearly 100% correctly. However, for these two model configurations, there were high miss-classifications in the cases of classes 4, 5, and 9. As revealed by the confusion matrix in Figure 6.13 (c), all the incorrect classifications by MobileNet+Softmax were in the form of classifying an image from a class rather than 6 as class 6. As revealed by the confusion matrix in Figure 6.13 (d), ResNet+Softmax has mis–classified images from class 4 as class 3 or 6, images from class 5 as class 6, and images from class 9 as class 1, 6 or 8. Our results confirm with the general practice in deep learning community where Softmax is typically used during the training of a deep net and then replaced by a bespoke classifier such as SVC during the test stage for a better classification accuracy.

Table 6.7: Percentages of Images being Classified as Correct and Wrong Distracted Driving for All CNN Model Configurations in Experiment 2.

Class Label		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Classified as Correct Distraction	AlexNet + Softmax	99.72	99.86	100.00	99.72	99.71	99.86	100.00	98.62	99.34
	AlexNet + SVC	99.72	99.86	100.00	99.86	99.86	99.86	100.00	98.45	99.51
	ResNet + Softmax	91.89	97.89	94.37	81.37	67.44	99.15	99.83	96.56	59.70
	ResNet + SVC	99.72	99.58	99.86	99.58	99.28	99.86	99.83	98.28	99.18
	MobileNet + Softmax	61.88	32.21	87.09	85.99	75.07	100.00	81.88	23.06	80.92
	MobileNet + SVC	99.57	99.58	99.86	99.72	99.42	99.86	99.65	99.31	99.01
	NasNet + Softmax	99.72	99.58	99.73	99.58	98.85	99.58	99.83	97.93	99.01
	NasNet + SVC	99.72	99.58	99.73	99.58	98.70	99.58	99.83	98.45	99.18
Classified as Wrong Distraction	AlexNet + Softmax	0.14	0.14	0.00	0.28	0.29	0.14	0.00	1.38	0.33
	AlexNet + SVC	0.14	0.14	0.00	0.14	0.14	0.14	0.00	1.38	0.49
	ResNet + Softmax	8.11	2.11	5.63	18.63	31.99	0.85	0.17	3.44	39.47
	ResNet + SVC	0.28	0.42	0.14	0.28	0.29	0.14	0.17	1.38	0.66
	MobileNet + Softmax	38.12	67.51	12.09	14.01	23.78	0.00	17.25	75.22	18.26
	MobileNet + SVC	0.14	0.42	0.14	0.28	0.29	0.14	0.35	0.69	0.49
	NasNet + Softmax	0.00	0.42	0.00	0.28	0.14	0.42	0.17	1.89	0.82
	NasNet + SVC	0.00	0.42	0.00	0.28	0.14	0.42	0.17	1.55	0.82

For the final evaluation, the proposed models were applied to test dataset provided by Kaggle. Table 6.8 presents the Kaggle scores for all four models with the Softmax and the SVC, respectively. In terms of performance comparison between different models, it can be observed that NASNet outperformed the other three models (lowest log loss score), followed by ResNet and MobileNet in terms of ranking. AlexNet, on the other hand, had the worst performance (highest log loss score). The performance of all other models was expected and in line with the validation data results. Even though AlexNet performance over validation data was over 99%, it failed when it was subjected to unseen test datasets. This confirms the hypothesis that AlexNet architecture was not well- trained and there was overfitting during the training which caused the model to quickly converge to highest accuracy. On the other hand, NASNet training followed the ideal training curve with smooth convergence and results of NASNet over test data are indicator that model was well-trained. Talking about the performance of Softmax and SVC, the performance of the SVC was slightly better than Softmax. Thus, the overall best model in terms of performance in this experiment was NASNet with the SVC.

Table 6.8: Kaggle Scores for Models with Softmax and SVC Classifiers Over Test Dataset

	Softmax		**SVC**	
	Public Score	Private Score	Public Score	Private Score
AlexNet	0.98153	1.02838	0.86610	0.918897
ResNet50	0.46246	0.51201	0.49651	0.44833
MobileNet	0.69831	0.64412	0.65180	0.63883
NASNet	0.34826	0.34235	0.34546	0.33957

Lastly, for the practical implementation, framerates and sizes of all four models were compared to evaluate the best choice for implementation in the real-world applications. In terms of sizes, no model was above 1GB size. This means that, given the state-of-the-art memories, all models can be easily implemented on standalone hardware. The exact sizes of the trained models were recorded as 251.2MB, 102MB, 90MB and 658MB for AlexNet,

ResNet, MobileNet and NASNet, respectively. Processing times or frame rates are considered one of the core requirements for a computer vision algorithm to be implemented in real-time. For a computer vision algorithm, it is important to be more accurate, but it is more vital to be fast enough to be implemented in real-time using standalone hardware. Table 6.9 presents the processing time for each model to predict a single instance of the test image into one of the ten output classes. From the processing time, it can be observed that MobileNet and AlexNet were the fastest, while NASNet was the slowest. The difference and importance of these measures is apparent. Before the processing time comparisons, based on the accuracies only, NASNet outperformed all other algorithms. However, for real-life implementation on low cost hardware, MobileNet is the valid choice.

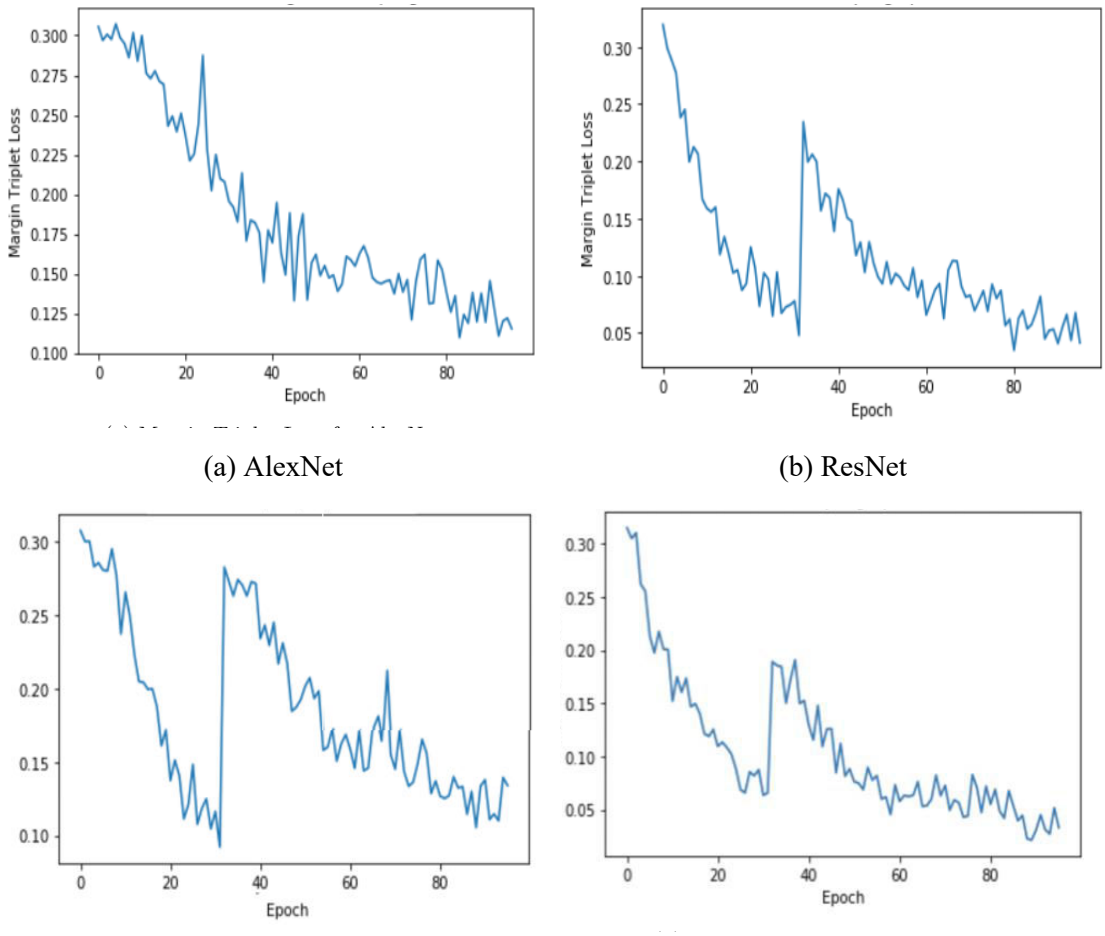
Table 6.9: Times for All Models with Softmax and SVC to Process Single Instance of Test Input.

	Processing Time for A Single Test Image	
	Softmax	**SVC**
AlexNet	4.82ms	4.79ms
ResNet50	8.84ms	8.91ms
MobileNet	4.94ms	4.72ms
NASNet	72.3ms	72.2ms

Results: We observe that the SVC as a classifier of the CNN features produced better result than the Softmax loss, it is able to generalize better between training and test data. Moreover, it fits the data in few seconds providing better training times than Softmax even with GPU acceleration over the CNN. The Table 6.4 shows the results obtained using both methods. Given the benefits of the SVC with the CNN features, we decided to further explore this approach, analysing how to obtain better features using triplet loss functions and how to further improve accuracy by proposing a novel modification to the triplet loss function.

6.6.3. Experiment 3 Results

Results of experiment set 2 highlighted the better performance of the SVC in comparison to the Softmax classifier for Kaggle distracted driving dataset. Based on the results of experiment set 2, the final experiments were planned in which performance of state-of-the-art deep architectures with the SVC was evaluated and compared for margin, naïve and batch triplet functions.



(a) AlexNet (b) ResNet

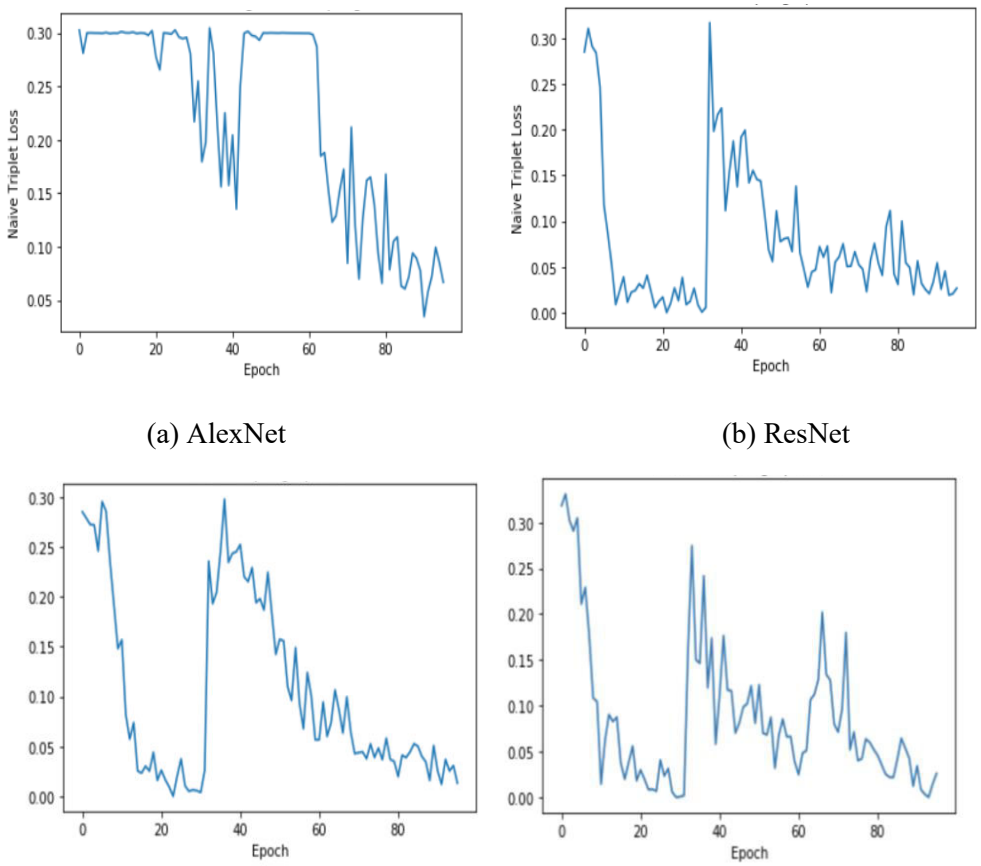
(c) MobileNet (d) NASNet

Figure 6.15: Training Loss Plots for All Models with Margin Triplet Loss.

At training time, we can observe how loss increases every time the training becomes more difficult. At epoch 32 we change to semi-hard negative mining and at epoch 64 we change to hard-negative mining except for the AlexNet diagram (top-left) that uses only random negative samples. The curve looks smoother in the Alexnet training. However, this does

not mean a better performance since the loss function in the other models is working with harder samples than with Alexnet. Figure 6.15 shows the training curves of the four architectures.

All the four deep models were trained over 70% of the Kaggle training dataset, and performances were plotted in terms of training loss plots for margin, naïve and batch triplet loss functions. Figure 6.15 presents the training loss plots for the models with margin triplet loss. Peaks in the training plots at epochs 32 and 64 are the indicators of shift of triplet mining from random to semi-hard and then to hard. Overall, the loss curves followed the standard training behaviour i.e. loss reduced with an increase in training epochs. In terms of performance, NASNet illustrated more smooth transition from semi-hard towards the hard triplet mining which is the indicator of a better and more stable model training.

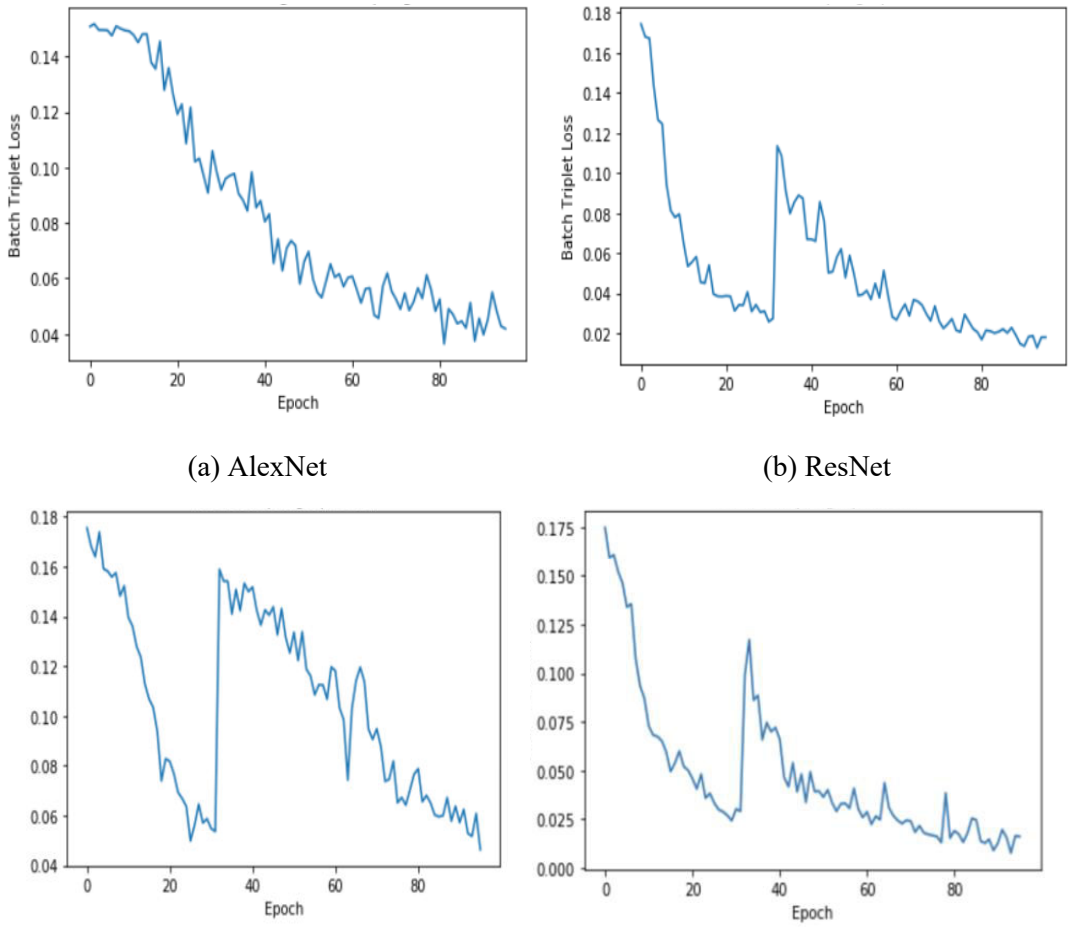


(a) AlexNet (b) ResNet

(c) MobileNet (d) NASNet

Figure 6.16: Training Loss Plots for All Model with Naïve Triplet Loss.

Figure 6.16 presents the loss plots for all the four models with the naïve triplet loss function implementation. The same training procedure was adopted as in the case of the margin triplet loss with transition from random to semi-hard at epoch 32 and then to hard triplet mining at epoch 64. From the loss plots, the peaks at the 32 and 64 epochs are the indicator of the shift between different triplet mining methods. Based on the plots, it can clearly be seen that there are a lot of ups and downs in the loss plots and the curves are not smooth, indicating a poor training of models. In terms of training performance, MobileNet was on the top, followed by ResNet and NASNet. Finally, Figure 6.17 presents the loss plots of all four models for the batch triplet loss function implementation. Loss plots were smooth in all cases, indicating a better training of models.



(a) AlexNet

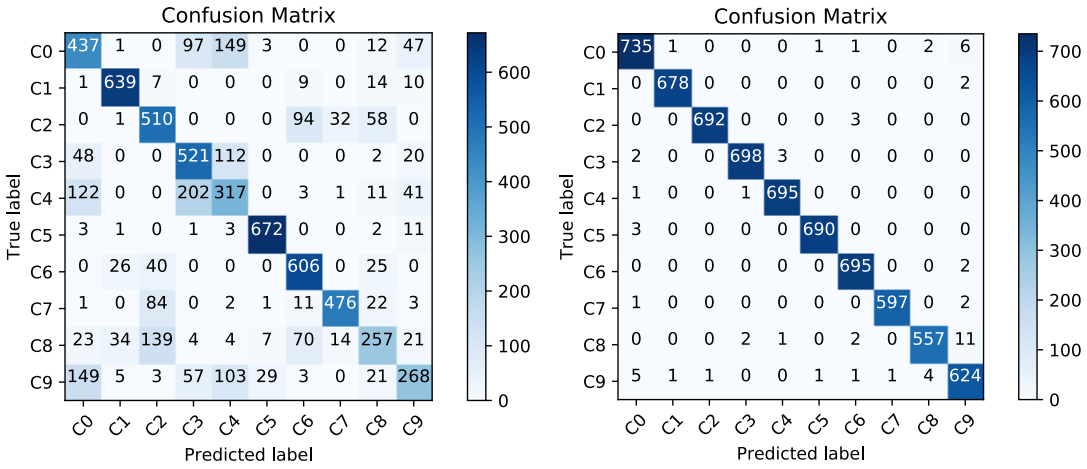
(b) ResNet

(c) MobileNet

(d) NASNet

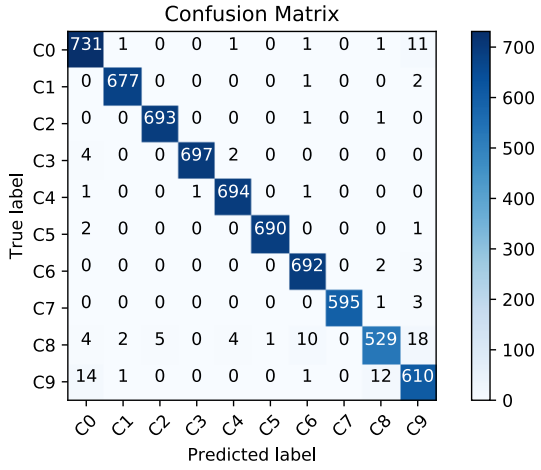
Figure 6.17: Training Loss Plots for All Models with Batch Triplet Loss.

In the second phase of evaluation, after the models were trained, the performance was evaluated over the validation data, and predictions were plotted as confusion matrices. Figure 6.18, Figure 6.19 and Figure 6.20 presents the confusion matrices of all four models for margin, naïve and batch triplet loss functions, respectively. From the confusion matrices, it can be observed that AlexNet performance was way below par as expected from the training phase since only random triplet mining was used. For the other three models, similar trend in terms of performance was observed as was in training process.

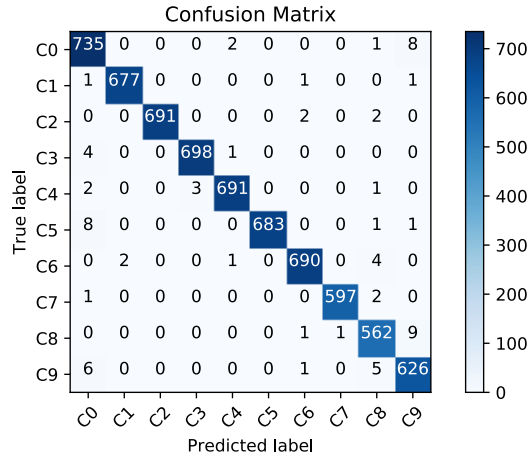


(a) AlexNet

(b) ResNet

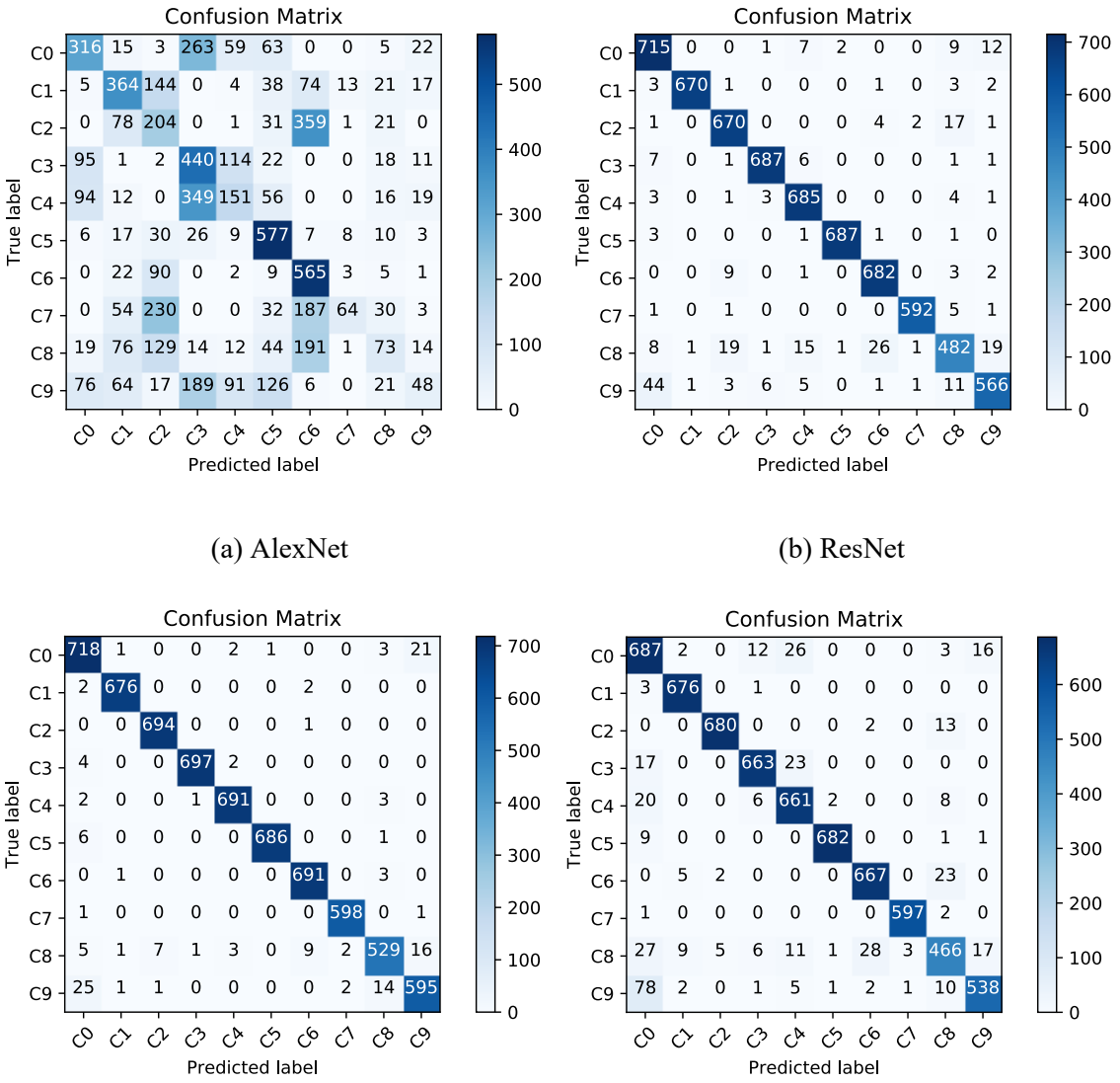


(c) MobileNet



(d) NASNet

Figure 6.18: Confusion Matrices for Models with Margin Triplet Loss Over Validation Dataset.



(a) AlexNet

(b) ResNet

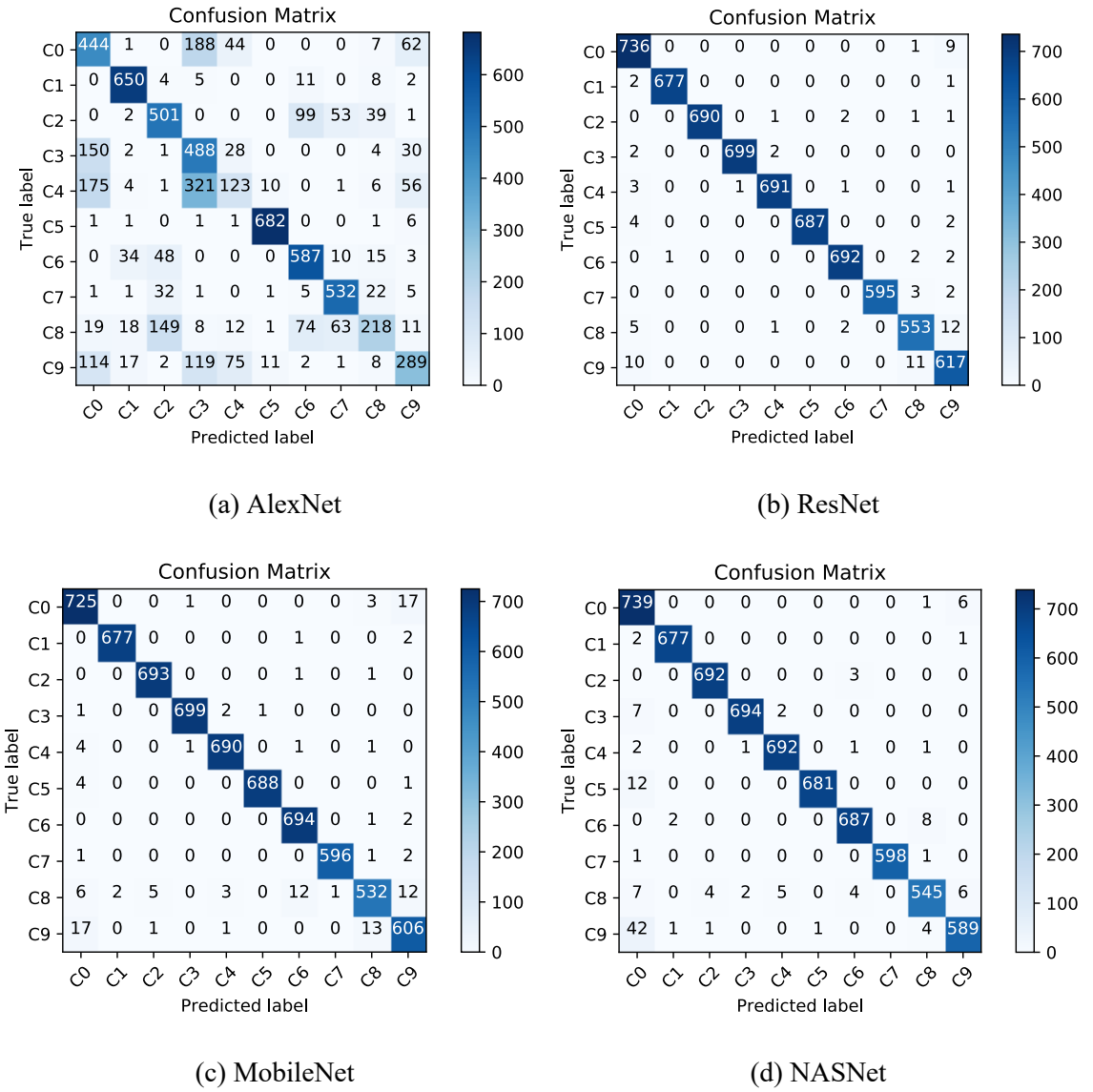
(c) MobileNet

(d) NASNet

Figure 6.19: Confusion Matrices for Models with Naïve Triplet Loss Over Validation Dataset.

Table 6.10 presents the comparison in terms of numerical values of training accuracies and validation accuracies of all four models with margin, naïve and batch triplet loss functions. From the table, it can clearly be observed that for margin triplet loss function, ResNet ranked first, NASNet ranked second, MobileNet ranked third with a negligible performance difference among the three models and AlexNet ranked the last with a nearly 30% drop in accuracy. For the naïve triplet loss function, MobileNet ranked first, ResNet ranked second,

NASNet ranked third with a small difference of less than 5% in accuracy whereas AlexNet ranked the last with a over 50% drop in accuracy. Finally, for the batch triplet loss function, ResNet ranked first, MobileNet ranked second, NASNet ranked third with a negligible difference in prediction accuracy and AlexNet ranked the last with a over 30% drop.



(a) AlexNet

(b) ResNet

(c) MobileNet

(d) NASNet

Figure 6.20: Confusion Matrices for Models with Batch Triplet Loss Over Validation Dataset.

From the triplet margin loss function confusion matrix in Figure 6.18, we can observe how the features produced by the AlexNet model are not accurate therefore the classification of the SVC becomes poor obtaining accuracies lower than 50% for the class 4. Overall, the

linear SVC with the features extracted from AlexNet does not perform well. The other three architectures are capable of producing high-quality embeddings using the adaptive triplet mining. The SVC with the linear kernel is capable of fitting the training data and having really good accuracy rates. However, MobileNet architecture is designed to operate with constrained hardware requirements. This has made it less powerful than other models like ResNet or NasNet.

As shown in Figure 6.19 and Figure 6.20, when using the features of the models trained with the naive triplet loss or the batch triplet loss function to train the SVC, we can see exactly the same pattern in the performance of the models as with margin triplet loss functions (in the confusion matrices of Figure 6.18). AlexNet performed the worst whereas the other three models produced comparable results.

Table 6.10: Numerical Comparison of All Models for Training and Validation Accuracies.

	Margin Triplet Loss		**Naïve Triplet Loss**		**Batch Triplet Loss**	
	Training Accuracy	Validation Accuracy	Training Accuracy	Validation Accuracy	Training Accuracy	Validation Accuracy
AlexNet	0.7039	0.6996	0.4261	0.4168	0.6779	0.6715
ResNet50	0.9955	0.9909	0.9685	0.9574	0.9917	0.9874
MobileNet	0.9908	0.9762	0.9876	0.9781	0.9887	0.9818
NASNet	0.9941	0.9893	0.9449	0.9397	0.9836	0.9809

Table 6.11 and Table 6.12 present the validation results of all the CNN model configurations in experiment set 3, where each model has been used with a SVC. As emphasised earlier, it is important to study that how many percent of images were classified as safe driving from the distracted driving classes since it is dangerous and will result in fatal consequences. Table 6.11 presents the percentages of images being classified as safe driving from all the distracted driving classes. From the table, it can be observed that AlexNet with all three configurations of margin, naïve and batch triplet loss performed

worst among all models in the sense that a large proportion of images of safe driving has been classified wrongly as distracted. As mentioned, AlexNet was trained using only random triplet mining while other models were trained using random, semi-hard and hard negative mining. In comparison to preliminary experiment, results of AlexNet in experiment 3 were improved because AlexNet model pre-trained over ImageNet dataset was used with SVC rather than Kaggle dataset trained AlexNet with Softmax as in preliminary experiment. Although overall for safe driving class, models with batch triplet loss function were able to give highest classification percentage, in terms of miss-classification as safe from other distraction classes margin triplet loss function was able to achieve the least number of wrong classifications. Highest percentages of miss-classification as safe driving were observed from class 9 which is consistent with the observations in preliminary experiments due to high similarity between class 0 and class 9.

Table 6.12 presents when images are classified as distracted driving, how many percent of them has been classified to the correct distraction class and how many wrong. As expected, AlexNet produced some errors. On the other hand, all the other three state-of-the-art models were able to achieve on average correct classifications above 90%. These results indicate that these state-of-the-art models have managed to learn and extract discriminative features and the SVC was able to correctly differentiate them into respective distracted driving classes.

In the third phase of evaluation, all the models were subjected to the Kaggle test dataset and predictions were recorded in excel sheet in a csv format. Prediction results for test evaluations were submitted to Kaggle. Table 6.13 presents the summary of all the log loss scores obtained from Kaggle website for the test dataset. From the table, it can be observed that once again NASNet outperformed all the other models in all three triplet loss implementations with the lowest log loss scores. Performance of AlexNet model was the worst among all as expected from the results of training and validation evaluations. Mixed performance with less difference was observed in ResNet and MobileNet models with ResNet outperforming MobileNet in batch triplet loss implementation while MobileNet outperforming ResNet in margin and naïve triplet loss implementations. Overall, batch triplet loss implementation with the NASNet model was the best in terms of performance

for the test dataset. More or less similar performances were recorded for MobileNet and ResNet models with triplet loss function implementations.

Table 6.11: Percentages of Images being Classified as Safe and Distracted Driving for All CNN Model Configurations in Experiment 3.

Class Label		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
		True +ev	**False -ev**								
Classified as Safe Driving	AlexNet + Margin	58.58	0.15	0.00	6.83	17.50	0.43	0.00	0.17	4.01	23.35
	AlexNet + Naïve	42.36	0.74	0.00	13.51	13.49	0.87	0.00	0.00	3.32	11.91
	AlexNet + Batch	59.52	0.00	0.00	21.34	25.11	0.14	0.00	0.17	3.32	17.87
	ResNet + Margin	98.53	0.00	0.00	0.28	0.14	0.43	0.00	0.17	0.00	0.78
	ResNet + Naïve	95.84	0.44	0.14	1.00	0.43	0.43	0.00	0.17	1.40	6.90
	ResNet + Batch	98.66	0.29	0.00	0.28	0.43	0.58	0.00	0.00	0.87	1.57
	MobileNet + Margin	97.99	0.00	0.00	0.57	0.14	0.29	0.00	0.00	0.70	2.19
	MobileNet + Naïve	96.25	0.29	0.00	0.57	0.29	0.87	0.00	0.17	0.87	3.92
	MobileNet + Batch	97.18	0.00	0.00	0.14	0.57	0.58	0.00	0.17	1.05	2.66
	NASNet + Margin	98.53	0.15	0.00	0.57	0.29	1.15	0.00	0.17	0.00	0.94
	NASNet + Naïve	92.09	0.44	0.00	2.42	2.87	1.30	0.00	0.17	4.71	12.23
	NASNet + Batch	99.06	0.29	0.00	1.00	0.29	1.73	0.00	0.17	1.22	6.58
		False +ev	**True -ev**								
Classified as Distracted Driving	AlexNet + Margin	41.42	99.85	100	93.17	82.50	99.57	100	99.83	95.99	76.65
	AlexNet + Naïve	57.64	99.26	100	86.49	86.51	99.13	100	100	96.68	88.09
	AlexNet + Batch	40.48	100	100	78.66	74.89	99.86	100	99.83	96.68	82.13
	ResNet + Margin	1.47	100	100	99.72	99.86	99.57	100	99.83	100	99.22
	ResNet + Naïve	4.16	99.56	99.86	99.00	99.57	99.57	100	99.83	98.60	93.10
	ResNet + Batch	1.34	99.71	100	99.72	99.57	99.42	100	100	99.13	98.43
	MobileNet + Margin	2.01	100	100	99.43	99.86	99.71	100	100	99.30	97.81
	MobileNet + Naïve	3.75	99.71	100	99.43	99.71	99.13	100	99.83	99.13	96.08
	MobileNet + Batch	2.82	100	100	99.86	99.43	99.42	100	99.83	98.95	97.34
	NASNet + Margin	1.47	99.85	100	99.43	99.71	98.85	100	99.83	100	99.06
	NASNet + Naïve	7.91	99.56	100	97.58	97.13	98.70	100	99.83	95.29	87.77
	NASNet + Batch	0.94	99.71	100	99.00	99.71	98.27	100	99.83	98.78	93.42

Table 6.12: Percentages of Images being Classified as Correct and Wrong Distracted Driving for All CNN Model Configurations in Experiment 3.

Class Label		1	2	3	4	5	6	7	8	9
Classified as Correct Distraction	AlexNet + Margin	93.97	73.38	74.11	45.48	96.97	86.94	79.33	44.85	42.01
	AlexNet + Naïve	53.53	29.35	62.59	21.66	83.26	81.06	10.67	12.74	7.52
	AlexNet + Batch	95.59	72.09	69.42	17.65	98.41	84.22	88.67	38.05	45.30
	ResNet + Margin	99.71	99.57	99.29	99.71	99.57	99.71	99.50	97.21	97.81
	ResNet + Naïve	98.53	96.40	97.72	98.28	99.13	97.85	98.67	84.12	88.71
	ResNet + Batch	99.56	99.28	99.43	99.14	99.13	99.28	99.17	96.51	96.71
	MobileNet + Margin	99.56	99.71	99.15	99.57	99.57	99.28	99.33	92.32	95.61
	MobileNet + Naïve	99.41	99.86	99.15	99.14	98.99	99.42	99.67	92.32	93.26
	MobileNet + Batch	99.56	99.71	99.43	99.00	99.28	99.57	99.33	92.84	94.98
	NASNet + Margin	99.56	99.42	99.29	99.14	98.56	99.00	99.50	98.08	98.12
	NASNet + Naïve	99.41	97.84	94.31	94.84	98.41	95.70	99.50	81.33	84.33
	NASNet + Batch	99.56	99.57	98.72	99.28	98.27	98.57	99.67	95.11	92.32
Classified as Wrong Distraction	AlexNet + Margin	5.88	26.62	19.06	37.02	2.60	13.06	20.50	51.13	34.64
	AlexNet + Naïve	45.74	70.65	23.90	64.85	15.87	18.94	89.33	83.94	80.56
	AlexNet + Batch	4.41	27.91	9.25	57.25	1.44	15.78	11.17	58.64	36.83
	ResNet + Margin	0.29	0.43	0.43	0.14	0.00	0.29	0.33	2.79	1.41
	ResNet + Naïve	1.03	3.45	1.28	1.29	0.43	2.15	1.17	14.49	4.39
	ResNet + Batch	0.15	0.72	0.28	0.43	0.29	0.72	0.83	2.62	1.72
	MobileNet + Margin	0.44	0.29	0.28	0.29	0.14	0.72	0.67	6.98	2.19
	MobileNet + Naïve	0.29	0.14	0.28	0.57	0.14	0.58	0.17	6.81	2.82
	MobileNet + Batch	0.44	0.29	0.43	0.43	0.14	0.43	0.50	6.11	2.35
	NASNet + Margin	0.29	0.58	0.14	0.57	0.29	1.00	0.33	1.92	0.94
	NASNet + Naïve	0.15	2.16	3.27	2.30	0.29	4.30	0.33	13.96	3.45
	NASNet + Batch	0.15	0.43	0.28	0.43	0.00	1.43	0.17	3.66	1.10

Table 6.13: Kaggle Scores for Models with Margin, Naïve and Batch Triplet Loss Functions Over Test Dataset.

	Margin Triplet Loss		**Naïve Triplet Loss**		**Batch Triplet Loss**	
	Public	Private	Public	Private	Public	Private
AlexNet	1.6068	1.6799	1.8928	1.9110	1.5638	1.5513
ResNet50	0.4407	0.5229	0.7334	0.6837	0.4475	0.4289
MobileNet	0.5836	0.5216	0.5707	0.5016	0.5311	0.5005
NASNet	0.3736	0.3149	0.5975	0.4664	0.3764	0.3168

Table 6.14: Times for All Models with Margin, Naïve and Batch Triplet Loss Functions to Process Single Instance of Test Input.

	Processing Time for Single Test Instance		
	Margin Triplet Loss	Naïve Triplet Loss	Batch Triplet Loss
AlexNet	8.7ms	8.6ms	8.5ms
ResNet50	7.57ms	7.38ms	7.61ms
MobileNet	4.69ms	5.9ms	4.65
NASNet	75.1ms	77.4ms	76.4ms

Finally, the performance in terms of processing times by each model was evaluated to analyse the practical implementation of algorithms on standalone hardware systems to perform real-time classification tasks. The processing time reported in this thesis were obtained from highly parallel and powerful machines. Therefore, they are not the real reflection of real-world performance. However, they were included to highlight the trend which would be same. When used on the low performance hardware, only the numbers will change. Table 6.14 presents the processing time taken by each model to process a single

instance of the test image. From the table, it can be observed that NASNet is the slowest among them, while MobileNet is fastest among them. The same results have been observed in experiment set 2. The importance of the practical implementation of models can be highlighted from this comparison. Although NASNet is the best among them in terms of prediction accuracy, it is the slowest by a huge margin. Hence, it is not the best choice for real-world implementation on hardware such as a portable device. On the other hand, the prediction accuracy from MobileNet was not poor compared to NASNet. Nevertheless, in terms of speed, it is almost 20 times faster, thus making it the best choice for the real-world implementation on portable devices.

6.7. Conclusion

Experiments performed in this research resulted into a number of insights. From the results of experiment set 2, it has been observed that SVC is much more powerful in terms of classification as compared to Softmax. From experiment set 3, it has been concluded that the novel batch triplet loss outperformed all others because it minimises intra-class variations as well as maximises inter-class variations. Furthermore, in terms of the best performed model, NASNet was the best for its accuracy while MobileNet was observed as the best for its lowest requirement on storage space and processing time without compromising the accuracy to a large extent. Overall, it can be concluded that if classification accuracy is the priority, NASNet model trained with margin triplet loss or batch triplet loss using SVC is the best in terms of performance when compared to AlexNet, MobileNet and ResNet. However, if processing time or real-time implementation on portable devices is the first priority, in that case MobileNet trained with batch triplet loss using SVC is the best in terms of performance when compared with other three addressed in this research.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1. Conclusion

Classification is a long-standing problem, with many different approaches to it. Focusing on the image domain, it was traditionally solved using hand-crafted features that were classified by a Multi-Layer Perceptron (MLP), SVM or other simpler methods like k-NN [156, 157]. Nowadays, the process of engineering hand-crafted features has been replaced by an end-to-end learning using CNNs [73, 158]. They take the raw image pixels as the input and are trained to learn the most discriminative features for a given classification task.

To improve road safety and a number of accidents due to hazardous behaviours such as drowsiness and distraction of drivers, this research has successfully addressed some research questions and investigated the potential solutions. To answer RQ1, a comprehensive subject review was performed, and the most relevant literature regarding local and deep learning approaches for distraction and drowsiness detection was reviewed. The literature was critically analysed and compared based on key factors in order to identify the obstacles to the real-world implementation of robust drowsiness and distraction detection systems. Some potential challenges identified as a result included the impact of lightning conditions on the accuracy of computer vision algorithms, the dependence on the performance of individual computer vision approaches involved, extensive computational power requirements, transition from high resolution to lower resolution detections, cost of system compared to vehicles, consequences of false alarms, visibility of regions of interest, early stage detections, and consideration of in-vehicle factors such as interference due to vibrations.

As for RQ2, conventional computer vision approaches such as detection of eye blinking, yawning and head pose were successfully implemented for small datasets to evaluate the performance.

To address RQ3, the proposed system was evaluated experimentally with a number of images taken from various standard databases. Given the use of smaller datasets, performance was evaluated using the k-fold cross validation approach in which whole dataset was used for training and testing. Classification accuracies of 99%, 98.5% and 99% were recorded for head pose, yawning and eye blinking detection, respectively. From the results over smaller datasets, a satisfactory performance was observed. However, in the conventional computer vision approaches, features were manually identified for each RI, such as the eyes, the mouth and the head. Furthermore, the conventional methods require a close-up image of the face/head to deliver a satisfactory performance. When the image resolution is low or when the face/head moves out of the camera view, conventional methods become powerless.

Talking about RQ4, the use of deep learning approaches is considered in the detection of distracted driving. Unlike conventional approaches where features are hand engineered to facilitate classification, deep learning approaches can be trained to learn the most discriminative features for a given classification task and the deep net architectures are able to learn such features from various regions of a human body simultaneously.

In reference to RQ5, various deep learning approaches were evaluated for their training performance, validation performance and test performance. Based on the encouraging results of preliminary experiments, in experiment 2, Softmax and the SVC classifier were compared for four states of the art deep models: AlexNet, ResNet, MobileNet and NASNet. From the results, the SVC performance was slightly better compared to that of Softmax. Finally, in experiment 3, the performance of all four deep architectures with the SVC was evaluated for margin, naïve and batch triplet loss implementations. The approach of transition from random to semi-hard to hard triplet mining was effectively used during the training of models. From the results, the performance of NASNet model with batch triplet loss implementation was ranked first in terms of accuracies; however, it was the slowest among them. The performance of ResNet and MobileNet was more or less similar to that of MobileNet in terms of fast processing times. The performance comparison between NASNet and MobileNet was not huge in terms of classification accuracies; however,

MobileNet was almost 20 times faster than NASNet, thus making MobileNet the best possible choice among them for real-world implementation on portable devices.

7.2. Future Work

In this MPhil thesis critical analysis of the literature have been conducted, which identifies potential challenges that occurs in the real-world implementation of computer vision methods in the detection of distraction driving. Each identified challenge can be a future direction to the research presented in this thesis. Listed are the possible future work for this research.

- Evaluation and improvement of existing algorithms for varied lighting conditions, more precisely, real in-vehicle lighting conditions especially during night time.
- Addressing and minimising the computational resources required to implement the proposed algorithms in real-world situation. One approach can be the use of image frames with a lower resolution to reduce the computational cost while keeping the accuracy up to a satisfactory level.
- Addressing the issue of false alarming to make it more user friendly. A system alarming the driver after every short interval of distraction is not practical since it will irritate the driver.
- The class 0 (safe driving) of Kaggle dataset does not include images of a driver checking a driving assistance equipment such as back mirror, side mirrors etc. Further experiments are required to ensure such images are not classified as distracted driving.
- Integration of non-vision based solutions to assist the scenarios when the driver's face is occluded.
- Incorporate the impact of vibrations and driving interferences on the detection accuracies.
- Implementation of hybrid conventional and deep learning\approaches to detect hazardous behaviours in drivers.

- Implementation of deep learning approaches on standalone hardware to test the real-world performances.

REFERENCES

[1] C. J. Murray and A. D. Lopez, "Alternative projections of mortality and disability by cause 1990–2020: Global Burden of Disease Study," *The Lancet,* vol. 349, no. 9064, pp. 1498-1504, 1997.

[2] P. K. Arnold, L. R. Hartley, A. Corry, D. Hochstadt, F. Penna, and A. M. Feyer, "Hours of work, and perceptions of fatigue among truck drivers," *Accident Analysis & Prevention,* vol. 29, no. 4, pp. 471-477, 1997.

[3] P. Philip *et al.*, "Fatigue, sleep restriction and driving performance," *Accident Analysis & Prevention,* vol. 37, no. 3, pp. 473-478, 2005.

[4] D. J. Beirness, H. M. Simpson, and A. Pak, *The road safety monitor: driver distraction*. Traffic Injury Research Foundation, 2002.

[5] E. Wahlstrom, O. Masoud, and N. Papanikolopoulos, "Vision-based methods for driver monitoring," in *Intelligent Transportation Systems, 2003. Proceedings. 2003 IEEE*, 2003, vol. 2: IEEE, pp. 903-908.

[6] K. Young, M. Regan, and M. Hammer, "Driver distraction: A review of the literature," *Distracted Driving,* pp. 379-405, 2007.

[7] J. C. Stutts, D. W. Reinfurt, L. Staplin, and E. A. Rodgman, "The role of driver distraction in traffic crashes," ed: Report prepared for AAA Foundation for Traffic Safety, Washington, DC, 2001.

[8] J.-S. Wang, R. R. Knipling, and M. J. Goodman, "The role of driver inattention in crashes: New statistics from the 1995 Crashworthiness Data System," in *40th annual proceedings of the Association for the Advancement of Automotive Medicine*, 1996, vol. 377, p. 392.

[9] A. H. Taylor and L. Dorn, "Stress, fatigue, health, and risk of road traffic accidents among professional drivers: the contribution of physical inactivity," *Annu. Rev. Public Health,* vol. 27, pp. 371-391, 2006.

[10] J. Park, Y. Kim, H. K. Chung, and N. Hisanaga, "Long working hours and subjective fatigue symptoms," *Industrial health,* vol. 39, no. 3, pp. 250-254, 2001.

[11] F. Sagberg, P. Jackson, H.-P. Krüger, A. Muzet, and A. Williams, *Fatigue, sleepiness and reduced alertness as risk factors in driving*. Institute of Transport Economics Oslo, 2004.

[12] S. K. Lal and A. Craig, "A critical review of the psychophysiology of driver fatigue," *Biological psychology,* vol. 55, no. 3, pp. 173-194, 2001.

[13] F. M. Carrier, "Regulatory Impact Analysis," 2011.

[14] M. Bayly, B. Fildes, M. Regan, and K. Young, "Review of crash effectiveness of Intelligent Transport Systems," *Emergency,* vol. 3, p. 14, 2006.

[15] D. Dinges and M. Mallis, "Managing fatigue by drowsiness detection: Can technological promises be realized?," in *INTERNATIONAL CONFERENCE ON FATIGUE AND TRANSPORTATION, 3RD, 1998, FREMANTLE, WESTERN AUSTRALIA*, 1998.

[16] H. Thimbleby, P. Duquenoy, and G. Marsden, "Ethics and consumer electronics," in *Proceedings of the 4th ETHICOMP International Conference on the Social and Ethical Impacts of Information and Communication Technologies—Ethicomp'99*, 1999.

[17] Kaggle. *Can computer vision spot distracted drivers?* [Online]. Available: https://www.kaggle.com/c/state-farm-distracted-driver-detection.

[18] J. Horne and L. Reyner, "Vehicle accidents related to sleep: a review," *Occupational and environmental medicine,* vol. 56, no. 5, pp. 289-294, 1999.

[19] A. Eskandarian, R. Sayed, P. Delaigue, A. Mortazavi, and J. Blum, "Advanced driver fatigue research," FMCSA-RRR-07-001, 2007.

[20] T. Nakamura, A. Maejima, and S. Morishima, "Detection of driver's drowsy facial expression," in *Pattern Recognition (ACPR), 2013 2nd IAPR Asian Conference on*, 2013: IEEE, pp. 749-753.

[21] W. Zhang, B. Cheng, and Y. Lin, "Driver drowsiness recognition based on computer vision technology," *Tsinghua Science and Technology,* vol. 17, no. 3, pp. 354-362, 2012.

[22] T. A. Ranney, E. Mazzae, R. Garrott, and M. J. Goodman, "NHTSA driver distraction research: Past, present, and future," in *Driver distraction internet forum*, 2000, vol. 2000.

[23] I. Hajime, B. ATSUMI, U. Hiroshi, and M. AKAMATSU, "Visual distraction while driving: trends in research and standardization," *IATSS research,* vol. 25, no. 2, pp. 20-28, 2001.

[24] D. Haigney, "Mobile phones and driving: A literature review," *RoSPA, Birmingham,* 1997.

[25] D. Line, "The Mobile Phone Report: A report on the effects of using a hand-held and a hands-free mobile phone on road safety," ed: Direct Line Insurance. Croydon: United Kingdom, 2002.

[26] A. L. Glaze and J. M. Ellis, "Pilot study of distracted drivers," *Transportation Safety Training Center for Public Policy,* 2003.

[27] J. D. Fuletra and D. Bosamiya, "A Survey on Driver's Drowsiness Detection Techniques," *International Journal on Recent and Innovation Trends in Computing and Communication,* vol. 1, no. 11, 2013.

[28] D. Haigney and S. Westerman, "Mobile (cellular) phone use and driving: A critical review of research methodology," *Ergonomics,* vol. 44, no. 2, pp. 132-143, 2001.

[29] E. Farber, J. Foley, and S. Scott, "Visual attention design limits for ITS in-vehicle systems: The Society of Automotive Engineers standard for limiting visual distraction while driving," in *Transportation Research Board Annual General Meeting*, 2000: Washington DC USA, pp. 2-3.

[30] J. J. Jain and C. Busso, "Assessment of driver's distraction using perceptual evaluations, self assessments and multimodal feature analysis," in *5th Biennial Workshop on DSP for In-Vehicle Systems, Kiel, Germany*, 2011.

[31] M. Baumann, A. Keinath, J. F. Krems, and K. Bengler, "Evaluation of in-vehicle HMI using occlusion techniques: experimental results and practical implications," *Applied ergonomics,* vol. 35, no. 3, pp. 197-205, 2004.

[32] M. Baumann *et al.*, "Assessing driver distraction using occlusion method and peripheral detection task," 2003.

[33] M. Wooldridge, K. Bauer, P. Green, and K. Fitzpatrick, "Comparison of driver visual demand in test track, simulator, and on-road environments," *Ann Arbor,* vol. 1001, pp. 48109-2150, 1999.

[34] H. Gao, A. Yuce, and J.-P. Thiran, "Detecting emotional stress from facial expressions for driving safety," in *Image Processing (ICIP), 2014 IEEE International Conference on*, 2014: IEEE, pp. 5961-5965.

[35] O. Langner, R. Dotsch, G. Bijlstra, D. H. Wigboldus, S. T. Hawk, and A. Van Knippenberg, "Presentation and validation of the Radboud Faces Database," *Cognition and emotion,* vol. 24, no. 8, pp. 1377-1388, 2010.

[36] N. C. Ebner, M. Riediger, and U. Lindenberger, "FACES—A database of facial expressions in young, middle-aged, and older women and men: Development and validation," *Behavior research methods,* vol. 42, no. 1, pp. 351-362, 2010.

[37] E. Tadesse, W. Sheng, and M. Liu, "Driver drowsiness detection through HMM based dynamic modeling," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, 2014: IEEE, pp. 4003-4008.

[38] T. D'Orazio, M. Leo, C. Guaragnella, and A. Distante, "A visual approach for driver inattention detection," *Pattern Recognition,* vol. 40, no. 8, pp. 2341-2355, 2007.

[39] M. Saradadevi and P. Bajaj, "Driver fatigue detection using mouth and yawning analysis," *International Journal of Computer Science and Network Security,* vol. 8, no. 6, pp. 183-188, 2008.

[40] S. Singh and N. P. Papanikolopoulos, "Monitoring driver fatigue using facial analysis techniques," in *Intelligent Transportation Systems, 1999. Proceedings. 1999 IEEE/IEEJ/JSAI International Conference on*, 1999: IEEE, pp. 314-318.

[41] D. F. Dinges and R. Grace, "PERCLOS: A valid psychophysiological measure of alertness as assessed by psychomotor vigilance," *US Department of Transportation, Federal Highway Administration, Publication Number FHWA-MCRT-98-006,* 1998.

[42] L. M. Bergasa, J. M. Buenaposada, J. Nuevo, P. Jimenez, and L. Baumela, "Analysing driver's attention level using computer vision," in *Intelligent*

Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on, 2008: IEEE, pp. 1149-1154.

[43] T. Kanade, Y. Tian, and J. F. Cohn, "Comprehensive database for facial expression analysis," in *fg*, 2000: IEEE, p. 46.

[44] A. Dasgupta, A. George, S. Happy, and A. Routray, "A Vision-Based System for Monitoring the Loss of Attention in Automotive Drivers," *IEEE Trans. Intelligent Transportation Systems,* vol. 14, no. 4, pp. 1825-1838, 2013.

[45] Q. Ji, Z. Zhu, and P. Lan, "Real-time nonintrusive monitoring and prediction of driver fatigue," *IEEE transactions on vehicular technology,* vol. 53, no. 4, pp. 1052-1068, 2004.

[46] L. Lang and H. Qi, "The study of driver fatigue monitor algorithm combined PERCLOS and AECS," in *2008 International Conference on Computer Science and Software Engineering*, 2008, vol. 1: IEEE, pp. 349-352.

[47] M. Sacco and R. A. Farrugia, "Driver fatigue monitoring system using support vector machines," in *Communications Control and Signal Processing (ISCCSP), 2012 5th International Symposium on*, 2012: IEEE, pp. 1-5.

[48] L. M. Bergasa, J. Nuevo, M. A. Sotelo, R. Barea, and M. E. Lopez, "Real-time system for monitoring driver vigilance," *Intelligent Transportation Systems, IEEE Transactions on,* vol. 7, no. 1, pp. 63-77, 2006.

[49] X. Fan, B.-C. Yin, and Y.-F. Sun, "Yawning detection for monitoring driver fatigue," in *Machine Learning and Cybernetics, 2007 International Conference on*, 2007, vol. 2: IEEE, pp. 664-668.

[50] E. Vural, M. Cetin, A. Ercil, G. Littlewort, M. Bartlett, and J. Movellan, "Drowsy driver detection through facial movement analysis," in *Human–computer interaction*: Springer, 2007, pp. 6-18.

[51] B.-C. Yin, X. Fan, and Y.-F. Sun, "Multiscale dynamic features based driver fatigue detection," *International Journal of Pattern Recognition and Artificial Intelligence,* vol. 23, no. 03, pp. 575-589, 2009.

[52] M. J. Flores, J. M. Armingol, and A. de la Escalera, "Real-time warning system for driver drowsiness detection using visual information," *Journal of Intelligent & Robotic Systems,* vol. 59, no. 2, pp. 103-125, 2010.

[53] D. Liu, P. Sun, Y. Xiao, and Y. Yin, "Drowsiness detection based on eyelid movement," in *Education Technology and Computer Science (ETCS), 2010 Second International Workshop on*, 2010, vol. 2: IEEE, pp. 49-52.

[54] Z. Zhang and J. Zhang, "A new real-time eye tracking based on nonlinear unscented Kalman filter for monitoring driver fatigue," *Journal of Control Theory and Applications,* vol. 8, no. 2, pp. 181-188, 2010.

[55] S. Park and M. Trivedi, "Driver activity analysis for intelligent vehicles: issues and development framework," in *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, 2005: IEEE, pp. 644-649.

[56] Q. N. Nguyen, L. T. A. Tho, T. V. Van, H. Yu, and N. D. Thang, "Visual based drowsiness detection using facial features," in *International Conference on the Development of Biomedical Engineering in Vietnam*, 2017: Springer, pp. 723-727.

[57] H. A. Kholerdi, N. TaheriNejad, R. Ghaderi, and Y. Baleghi, "Driver's drowsiness detection using an enhanced image processing technique inspired by the human visual system," *Connection Science,* vol. 28, no. 1, pp. 27-46, 2016.

[58] J. Jo, S. J. Lee, H. G. Jung, K. R. Park, and J. Kim, "Vision-based method for detecting driver drowsiness and distraction in driver monitoring system," *Optical Engineering,* vol. 50, no. 12, pp. 127202-127202-24, 2011.

[59] A. Nabo, "Driver attention—Dealing with drowsiness and distraction," *Göteborg: IVSS,* 2009.

[60] L. Yunqi, Y. Meiling, S. Xiaobing, L. Xiuxia, and O. Jiangfan, "Recognition of eye states in real time video," in *Computer Engineering and Technology, 2009. ICCET'09. International Conference on*, 2009, vol. 1: IEEE, pp. 554-559.

[61] C. Craye and F. Karray, "Driver distraction detection and recognition using RGB-D sensor," *arXiv preprint arXiv:1502.00250,* 2015.

[62] Y. Liao, S. E. Li, G. Li, W. Wang, B. Cheng, and F. Chen, "Detection of driver cognitive distraction: An SVM based real-time algorithm and its comparison study

in typical driving scenarios," in *Intelligent Vehicles Symposium (IV), 2016 IEEE*, 2016: IEEE, pp. 394-399.

[63] K. Kircher, C. Ahlstrom, and A. Kircher, "Comparison of two eye-gaze based real-time driver distraction detection algorithms in a small-scale field operational test," in *Proc. 5th Int. Symposium on Human Factors in Driver Assessment, Training and Vehicle Design*, 2009, pp. 16-23.

[64] J. Pohl, W. Birk, and L. Westervall, "A driver-distraction-based lane-keeping assistance system," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering,* vol. 221, no. 4, pp. 541-552, 2007.

[65] E. Murphy-Chutorian, A. Doshi, and M. M. Trivedi, "Head pose estimation for driver assistance systems: A robust algorithm and experimental evaluation," in *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, 2007: IEEE, pp. 709-714.

[66] F. Chollet, *Deep learning with python*. Manning Publications Co., 2017.

[67] D. Xie, L. Zhang, and L. Bai, "Deep Learning in Visual Computing and Signal Processing," *Applied Computational Intelligence and Soft Computing,* vol. 2017, 2017.

[68] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation,* vol. 18, no. 7, pp. 1527-1554, 2006.

[69] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Deep neural networks segment neuronal membranes in electron microscopy images," in *Advances in neural information processing systems*, 2012, pp. 2843-2851.

[70] H. R. Roth *et al.*, "Improving computer-aided detection using convolutional neural networks and random view aggregation," *IEEE transactions on medical imaging,* vol. 35, no. 5, pp. 1170-1181, 2016.

[71] S. Xie and Z. Tu, "Holistically-nested edge detection," *International Journal of Computer Vision,* pp. 1-16, 2017.

[72] N. Tajbakhsh *et al.*, "Convolutional neural networks for medical image analysis: full training or fine tuning?," *IEEE transactions on medical imaging,* vol. 35, no. 5, pp. 1299-1312, 2016.

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

[74] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097-1105.

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

[76] Y. Taigman, M. Yang, M. A. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1701-1708.

[77] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 815-823.

[78] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer,* vol. 29, no. 3, pp. 31-44, 1996.

[79] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807-814.

[80] H. Drucker, C. J. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, "Support vector regression machines," in *Advances in neural information processing systems*, 1997, pp. 155-161.

[81] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*: Springer, 2010, pp. 177-186.

[82] R. M. Gray, *Entropy and information theory*. Springer Science & Business Media, 2011.

[83] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016.

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

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

[86] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167,* 2015.

[87] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research,* vol. 15, no. 1, pp. 1929-1958, 2014.

[88] A. Y. Ng, "Feature selection, L 1 vs. L 2 regularization, and rotational invariance," in *Proceedings of the twenty-first international conference on Machine learning*, 2004: ACM, p. 78.

[89] R. Caruana, S. Lawrence, and C. L. Giles, "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping," in *Advances in neural information processing systems*, 2001, pp. 402-408.

[90] K. Dwivedi, K. Biswaranjan, and A. Sethi, "Drowsy driver detection using representation learning," in *Advance Computing Conference (IACC), 2014 IEEE International*, 2014: IEEE, pp. 995-999.

[91] S. Park, F. Pan, S. Kang, and C. D. Yoo, "Driver Drowsiness Detection System Based on Feature Representation Learning Using Various Deep Networks," in *Asian Conference on Computer Vision*, 2016: Springer, pp. 154-164.

[92] B. Y. Zhang, S. University, Ed. *Apply and Compare Different Classical Image Classification Method: Detect Distracted Driver*. 2016.

[93] H. W. Rendani Mbuvha, "Convolutional Neural Networks for Distracted Driver Detection," KTH Royal Institute of Technology, 2016.

[94] M. Venturelli, G. Borghi, R. Vezzani, and R. Cucchiara, "Deep Head Pose Estimation from Depth Data for In-car Automotive Applications," *arXiv preprint arXiv:1703.01883,* 2017.

[95] C. Streiffer, R. Raghavendra, T. Benson, and M. Srivatsa, "Darnet: a deep learning solution for distracted driving detection," in *Proceedings of the 18th*

ACM/IFIP/USENIX Middleware Conference: Industrial Track, 2017: ACM, pp. 22-28.

[96] M. R.-S. Flora Dellinger, Erwan Bernard, Laurette Guyonvarch, Anne Guillaume, "Computer vision algorithms for detecting secondary tasks in naturalistic driving studies," presented at the 5th International Conference on Driver Distraction and Inattention, 2017.

[97] G. Masala and E. Grosso, "Real time detection of driver attention: Emerging solutions based on robust iconic classifiers and dictionary of poses," *Transportation research part C: emerging technologies,* vol. 49, pp. 32-42, 2014.

[98] Y. Abouelnaga, H. M. Eraqi, and M. N. Moustafa, "Real-time Distracted Driver Posture Classification," *arXiv preprint arXiv:1706.09498,* 2017.

[99] M. D. Hssayeni, S. Saxena, R. Ptucha, and A. Savakis, "Distracted Driver Detection: Deep Learning vs Handcrafted Features," *Electronic Imaging,* vol. 2017, no. 10, pp. 20-26, 2017.

[100] T. Wang and P. Shi, "Yawning detection for determining driver drowsiness," in *VLSI Design and Video Technology, 2005. Proceedings of 2005 IEEE International Workshop on*, 2005: IEEE, pp. 373-376.

[101] M. S. B. Zainal, I. Khan, and H. Abdullah, "Efficient Drowsiness Detection by Facial Features Monitoring," *Research Journal of Applied Sciences, Engineering and Technology,* vol. 7, no. 11, pp. 2376-2380, 2014.

[102] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision,* vol. 57, no. 2, pp. 137-154, 2004.

[103] S. Lajevardi, "Automatic recognition of facial expressions," BEng, RMIT, 2011.

[104] S. Ghosh, T. Nandy, and N. Manna, "Real Time Eye Detection and Tracking Method for Driver Assistance System," in *Advancements of Medical Electronics*: Springer, 2015, pp. 13-25.

[105] P. Gejguš and M. Šperka, "Face tracking in color video sequences," in *Proceedings of the 19th spring conference on Computer graphics*, 2003: ACM, pp. 245-249.

[106] M. J. Flores, J. M. Armingol, and A. de la Escalera, "Driver drowsiness warning system using visual information for both diurnal and nocturnal illumination

conditions," *EURASIP journal on advances in signal processing,* vol. 2010, p. 3, 2010.

[107] S. Weijie, S. Haixin, E. Cheng, Z. Qingkun, Q. Li, and S. Weijun, "Effective Driver Fatigue Monitoring through Pupil Detection and Yawing Analysis in Low Light Level Environments," *International Journal of Digital Content Technology & its Applications,* vol. 6, no. 17, 2012.

[108] S. Abtahi, B. Hariri, and S. Shirmohammadi, "Driver drowsiness monitoring based on yawning detection," in *Instrumentation and Measurement Technology Conference (I2MTC), 2011 IEEE*, 2011: IEEE, pp. 1-4.

[109] B. Hariri, S. Abtahi, S. Shirmohammadi, and L. Martel, "A yawning measurement method to detect driver drowsiness," *Technical Papers,* 2012.

[110] H.-B. Kang, "Various approaches for driver and driving behavior monitoring: a review," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2013, pp. 616-623.

[111] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, 1992: ACM, pp. 144-152.

[112] Y.-S. Wu, T.-W. Lee, Q.-Z. Wu, and H.-S. Liu, "An eye state recognition method for drowsiness detection," in *Vehicular Technology Conference (VTC 2010-Spring), 2010 IEEE 71st*, 2010: IEEE, pp. 1-5.

[113] S. S. Haykin, S. S. Haykin, S. S. Haykin, and S. S. Haykin, *Neural networks and learning machines*. Pearson Education Upper Saddle River, 2009.

[114] M. B. Christopher, *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2016.

[115] D. Fradkin and I. Muchnik, "Support vector machines for classification," *Discrete methods in epidemiology,* vol. 70, pp. 13-20, 2006.

[116] K. P. Bennett and C. Campbell, "Support vector machines: hype or hallelujah?," *Acm Sigkdd Explorations Newsletter,* vol. 2, no. 2, pp. 1-13, 2000.

[117] N. Gourier and J. Letessier, "The pointing 04 data sets," in *Proceedings of Pointing 2004, ICPR International Workshop on Visual Observation of Deictic Gestures*, 2004, pp. 1-4.

[118] H. Ujir, M. Spann, and I. H. M. Hipiny, "3D facial expression classification using 3D facial surface normals," in *The 8th International Conference on Robotic, Vision, Signal Processing & Power Applications*, 2014: Springer, pp. 245-253.

[119] O. Jesorsky, K. J. Kirchberg, and R. W. Frischholz, "Robust face detection using the hausdorff distance," in *International conference on audio-and video-based biometric person authentication*, 2001: Springer, pp. 90-95.

[120] H. Saleem, "eye-blink-detection."

[121] O. Russakovsky *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision,* vol. 115, no. 3, pp. 211-252, 2015.

[122] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146,* 2016.

[123] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, 2017: IEEE, pp. 5987-5995.

[124] A. Veit, M. Wilber, and S. Belongie, "Residual networks are exponential ensembles of relatively shallow networks. arXiv preprint," *arXiv preprint arXiv:1605.06431,* vol. 1, no. 2, p. 3, 2016.

[125] M. Abdi and S. Nahavandi, "Multi-residual networks," *CoRR, abs/1609.05672,* vol. 8, 2016.

[126] X. Zhang, Z. Li, C. C. Loy, and D. Lin, "Polynet: A pursuit of structural diversity in very deep networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017: IEEE, pp. 3900-3908.

[127] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556,* 2014.

[128] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *AAAI*, 2017, vol. 4, p. 12.

[129] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818-2826.

[130] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861,* 2017.

[131] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," *arXiv preprint arXiv:1707.07012,* 2017.

[132] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering,* vol. 22, no. 10, pp. 1345-1359, 2010.

[133] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," in *Advances in neural information processing systems*, 1994, pp. 737-744.

[134] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 2005, vol. 1: IEEE, pp. 539-546.

[135] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML Deep Learning Workshop*, 2015, vol. 2.

[136] Y. LeCun *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural computation,* vol. 1, no. 4, pp. 541-551, 1989.

[137] M. Lagunas and E. Garces, "Transfer Learning for Illustration Classification," *arXiv preprint arXiv:1806.02682,* 2018.

[138] Y. Lin *et al.*, "Large-scale image classification: fast feature extraction and svm training," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 2011: IEEE, pp. 1689-1696.

[139] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1717-1724.

[140] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: an astounding baseline for recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014, pp. 806-813.

[141] D. Cheng, Y. Gong, S. Zhou, J. Wang, and N. Zheng, "Person re-identification by multi-channel parts-based cnn with improved triplet loss function," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1335-1344.

[142] C. Zhang and K. Koishida, "End-to-End Text-Independent Speaker Verification with Triplet Loss on Short Utterances," in *Interspeech*, 2017, pp. 1487-1491.

[143] X. Wang. (2016). *Tutorial: Triplet Loss Layer Design for CNN* [Online]. Available: http://www.cnblogs.com/wangxiaocvpr/p/5452367.html.

[144] J. Daugman, "How iris recognition works," *IEEE Transactions on circuits and systems for video technology,* vol. 14, no. 1, pp. 21-30, 2004.

[145] J. Daugman, "Biometric decision landscapes," University of Cambridge, Computer Laboratory, 2000.

[146] B. Kumar, G. Carneiro, and I. Reid, "Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5385-5394.

[147] W. Ding, R. Wang, F. Mao, and G. Taylor, "Theano-based large-scale visual recognition with multiple gpus," *arXiv preprint arXiv:1412.2302,* 2014.

[148] A. B. Graf and S. Borer, "Normalization in support vector machines," in *Joint Pattern Recognition Symposium*, 2001: Springer, pp. 277-282.

[149] BAIR. *Caffe: A Deep Learning Framework* [Online]. Available: http://caffe.berkeleyvision.org/.

[150] F. Chollet, "Keras," ed, 2015.

[151] M. U. s. Guide, "The mathworks," *Inc., Natick, MA,* vol. 5, p. 333, 1998.

[152] A. Paszke, S. Gross, S. Chintala, and G. Chanan, "PyTorch," ed, 2017.

[153] M. Abadi *et al.*, "Tensorflow: a system for large-scale machine learning," in *OSDI*, 2016, vol. 16, pp. 265-283.

[154] J. Bergstra *et al.*, "Theano: Deep learning on gpus with python," in *NIPS 2011, BigLearning Workshop, Granada, Spain*, 2011, vol. 3: Citeseer, pp. 1-48.

[155] R. Collobert, S. Bengio, and J. Mariéthoz, "Torch: a modular machine learning software library," Idiap, 2002.

[156] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 2009: IEEE, pp. 1794-1801.

[157] H. Zhang, A. C. Berg, M. Maire, and J. Malik, "SVM-KNN: Discriminative nearest neighbor classification for visual category recognition," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, 2006, vol. 2: IEEE, pp. 2126-2136.

[158] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature,* vol. 521, no. 7553, pp. 436-444, 2015.