TABLE OF CONTSENTS

ABSTRACT

1.0 INTRODUCTION

1.1 AIMS

1.2 OBJECTIVES

1.3 JUSTIFICATION

2.0 LITERATURE REVIEW

3.0 METHODOLOGY

3.1 PROPOSED METHODOLOGY

3.2 DATA SETS

3.3 TOOLS USED

3.4 MODEL DEVELOPMENT

3.5 TRAINING AND VALIDATION

3.6 MODEL TRAINING AND VALIDATION COMPLETION

4.0 RESULTS AND ANALYSIS

5.0 CONCLUSION AND RECOMMENDATION

LIST OF TABLES AND FIGURES

LIST OF ABBREVIATIONS

APPENDICES

APPENDIX I EYE DETECTION MODEL

APPENDIX II FACE DETECTION MODEL

APPENDIX III DATA TESTING MODEL

LIST OF TABLES AND FIGURES

*Figure 3.1 Methodology for the detection of driver drowsiness……………………………….*

*Figure 3.2 Eye Detection Training and Validation Accuracy and Loss Diagrams………………………*

*Figure 3.3 Face Detection Training and Validation Accuracy and Loss Diagram………………………*

*Figure 4.1 Precision-Recall Curve for Face Detection Model………………………………..*

*Figure 4.2 Precision-Recall Curve for Eye Detection Model………………………………………………*

*Figure 4.3 ROC Curve for Face Detection Model……………………………………………………………..*

*Figure 4.4 ROC Curve for Eye Detection Model………………………………………………………………*

*Table 3.1 A table summarizing the final augmentation techniques and their parameters……………..*

*Table 4.1 Accuracy Rates for Eye and Face Detection Models……………………………………………*

*Table 4.2 F1 Score Deduced from The Eye and Face Detection Model……………………………….*

*Table 4.3 Performance Metrics Summary……………………………………………...*

LIST OF ABBREVIATIONS

ABBREVIATION FULL MEANING

1-D LSTM One-dimensional Long Short-Term Memory network

AlexNet Alex Krizhevsky Network

BiLSTM Bidirectional Long Short-Term Memory

C-LSTM Convolutional Long Short-Term Memory

CNN Convolutional Neural Network

DLIB Data Library

EAR Eye Aspect Ratio

ECG Electrocardiography

EEG Electroencephalography

EOG Electrooculography

FLOWIMAGENET Flow Image Network

GPUs Graphical Processing Units

HRV Heart Rate Variability

INCEPTION-V3 Inception version 3

IR Infra- Red

LSTM Long Short-Term Memory

MAR Mouth Aspect Ratio

MRL Media Research Lab

NTHU National Tsuing Hua University

OpenCV Open-Source Computer Vision Library

PERCLOS Percentage of Eye Closure

PPGI Photoplethysmography imaging

ResNet Residual Network

RGB Red Green Blue

R-LSTM Recurrent Long Short-Term Memory

SoftMax Soft Maximum

SSD Single Shot Multibox Detector

UTA University of Texas at Arlington

VGG Visual Geometric Group

VGG16 Visual Geometric Group model 16

YOLOv3 You Only Look Once version 3

**ABSTRACT**

Across the globe, there is an increasing number of accidents and injuries (both fatal and non-fatal) occurring daily. Most of these accidents are attributed to drivers’ drowsiness because of many factors like exhaustion, long driving hours, medication, disease etc. This in part can be attributed to the substantial changes in the way that people manage their time in modern times, which has altered their sleep cycle. Driving is an activity that requires a sound and functional state of mind and with sleep cycle altered because of demanding work schedules, may result in drowsy driving. Irregular sleep habits and inadequate sleep results in falling asleep at various times of the day.As there is an increase in demand for mobility, safety measures need to be put in place to reduce the occurrence of these incidents. Detecting driver drowsiness is a crucial aspect of addressing driver fatigue, a significant contributor to road accidents. The identification of drowsiness allows for timely interventions and preventive measures, enhancing road safety. This study employs convolutional neural networks along with TensorFlow and Keras (two deep learning models) to create a model for detecting driver drowsiness, utilizing a webcam to capture images. The level of driver alertness will be assessed using PERCLOS. OpenCV is utilized for processing real-time video data to identify drowsiness indicators. The models' performance is evaluated using precision, accuracy, recall, and F1 score, adopting a comprehensive approach. Evaluation on the dataset revealed an impressive F1 score of 0.944882 for eye detection, indicating its effectiveness in discerning open and closed eye states, and 0.893617 for face detection. Testing involved a dataset comprising 17,775 images of eye closure, 17,762 images of eye openness, 15,837 images of drowsy facial conditions, and 12,937 images of non-drowsy facial conditions associated with driver sleepiness, encompassing various features like head position and illumination. The models exhibited a high level of accuracy, with the CNN model achieving 95.7% for eye detection and 92.3% for face detection, along with an average precision-recall of 90%. The Area Under the Receiver Operating Characteristic (ROC) Curve (AUC-ROC) yielded a modest result of 89%.

**CHAPTER ONE**

**1.0 INTRODUCTION**

Globally, the frequency of road accidents has surged, largely due to the growing number of vehicles congesting roadways, increasing the risk of collisions. In today's rapidly evolving world, where roads serve as crucial arteries of transportation, ensuring the safety of both drivers and passengers is paramount. Drowsy driving, identified as the second leading cause of accidents after alcohol consumption (Verwey and Zaidel, 1999: Wang et al., 2017). , has led to numerous single and multiple collisions resulting in significant fatalities, injuries (Bhandayker, 2019; Satish et al., 2023), financial losses, and extensive property damage worldwide (Shaik, 2023).

Fatigue behind the wheel poses a significant global concern, being a leading factor in road accidents, as highlighted by Singh et al. (2023). The World Health Organization (WHO, 2020) reported a staggering 1.35 million fatalities and 20-50 million non-fatal serious injuries worldwide due to road traffic accidents, a figure equivalent to the population of Maine. Cai et al. (2020) observed that in the United States, 21% of fatal motor vehicle crashes and 13% of severe injuries were attributed to drowsy driving, based on AAA foundation data. With a rising trend in accidents caused by drowsy driving, there is a pressing need to explore techniques for effectively detecting and predicting drowsiness early to enhance transportation safety (Shaik, 2023).

Drowsiness, also commonly referred to as sleepiness or tiredness, often results in falling asleep in an inappropriate situation and time (Arunasalam et al.,2020). Drowsy driving, also referred to as driver fatigue or tired driving, refers to the action of operating a motor vehicle while experiencing fatigue or sleepiness (Moradi, Nazari and Rahmani, 2019: Czeisler et al., 2016). This can be caused by job related stress, sleep deprivation, long driving hours, alcohol consumption, medication, medical illnesses, etc. Driver drowsiness detection is a technology designed to identify the extent of tiredness or drowsiness exhibited by a driver during vehicle operation. The objective of this technology is to notify the driver or implement precautionary actions, such as reducing speed or halting the vehicle, with the aim of averting accidents resulting from driver fatigue.

Some symptoms of driver drowsiness include yawning, blinking frequently, inability to focus like drifting from lane, difficulty keeping eyes open, difficulty concentrating, nodding, unjustifiable variations in speed, poor judgement, delayed reaction time etc. The combined impacts of these factors significantly impair performance, alertness, memory, focus, and reaction times. (Dernocoeur, 2000; Mahowald and Bornemann, 2006). Researchershave categorized driver drowsiness based on the followingfactors: Physiological, Vehicle-based, and Behavioral parameters (Dua et al., 2021).

In physiological measurement of drivers’ drowsiness, sensors, and electronic gadgets are attached to the drivers’ bodies or skin to measure their heart rate, pulse rate, brain activity, body temperature, and other physiological variables. Three signals used in this evaluation will include electroencephalography (EEG), electrooculography (EOG), and electrocardiography(ECG) to evaluate the state of the driver (Barua et al., 2019). Despite their ability to provide highly accurate results, these devices face limited acceptability due to their practical constraints.

Vehicle based drowsy detection relies on vehicle control system incorporating indicators like steering wheel actions, braking behaviors, the vehicles position on a lane etc. monitoring of car movement pattern is done through sensors installed to measure various vehicle and street parameters like steering wheel angle, speed, or deviation from the lane. Of the various methods listed, steering wheel measurements typically offer superior outcomes compared to alternative vehicle-centric approaches. Though the vehicle-based techniques are non-intrusive, their accuracy in detecting drowsiness varies and is highly dependent on factors like the condition of the road and how good a driver is.

Behavioral detection, on the other hand, is more dependable than vehicle-based measures, because it centers on the individual rather than the vehicle. It detects drowsiness by monitoring head movements and facial parameters such as the eyes, face expression, eyebrows etc. Head movement is captured by using a camera (Bamidele et al., 2019, Kiashari et al., 2019).

The huge negative impact caused by drowsy drivers has made researchers explore various methods of identifying drowsy driving. Many studies have explored various detection techniques for identifying drowsy drivers. Garcia et al., (2012), developed a non-intrusive approach to driver drowsiness detection using an IR camera positioned in front of the driver’s dashboard to detect his face and obtain drowsiness clues from their eyes closure. This system comprises of three stages; preprocessing, pupil position detection and use of PERCLOS to calculate eye closure information.

Chakraborty and Aoyon, (2014), suggested a fatigue detection system that utilizes Behavioral Studies with Computer Vision. It aims to offer a quick and straightforward detection technique. The system works by initially detecting faces, then locating eyes to analyze the existence of eye pupils and measureblink rate. Based on these parameters, the system determines the driver's fatigue level.

This research describes a simple, non-intrusive and affordable system to detect drowsy drivers in real-time using a webcam. It works by using image processing to locate the driver's face, then tracking the eyes when the face has been found and then checking for signs of drowsiness by analyzing features of the eyes (like how closed they are). The face is detected in each frame by employing image processing techniques using PERCLOS. Eyelid ratio is thenused to determine drowsiness and trigger warnings to alert the driver to avoid accidents, thereby saving lives and properties.

In addition to the introduction, the research paper will be structured into different sections. Chapter two will consist of an elaborate literature review, while chapter three will outline the methodology used in this study. Chapter four will present the results and analysis of our findings and chapter five will not only summarize the findings of the research but also discuss the contributions of this study to driver drowsy detection.

**1.1 RESEARCH AIMS**

To design and develop an efficient and accurate driver monitoring system using convolutional neural networks (CNNs) and deep learning techniques, leveraging webcam-based face detection, with the primary objective of enhancing road safety through real-time assessment of driver attentiveness. This research aims to explore the capabilities of CNNs in facial recognition and classification, optimizing model architecture, training procedures, and data augmentation techniques to achieve robust performance in detecting various states of driver alertness, including drowsiness, distraction, and impairment.

Furthermore, the study aims to investigate the feasibility and effectiveness of integrating the developed model into existing vehicle safety systems or deploying it as a standalone application for personal vehicles. Through empirical evaluation and analysis of collected data, this research seeks to contribute insights into improving road safety measures and reducing the incidence of accidents caused by driver fatigue or distraction.

**1.2 OBJECTIVES**

* To identify the key factors that lead to drowsiness among drivers.
* To highlight the impact of the increased cases of driver’s drowsiness on road safety.
* To find an automated approach for the detection of driver’s drowsiness.
* Extensive review of literature to analyze previous works in a bid to select the most appropriate technique in tackling the problem
* Data collection and preprocessing
* Feature extraction and selection
* Model design with AI-based technique convolutional Neural network and deep learning
* Model training and validation
* Implementation of designed model
* Testing of the designed model
* Evaluation of the model to ascertain if design objectives were met

**1.3 JUSTIFICATION**

In today's environment, where fatigue frequently leads to accidents resulting in fatalities and serious injuries, it is critical to develop steps to limit this risk. This may be accomplished through research focused on developing technology that can minimise traffic accidents, hence improving road safety. Building a driver detection system that uses a convolutional neural network (CNN) in conjunction with webcam-based face identification enables continuous real-time evaluation of the driver's attentiveness without the need for specialised hardware, which are very expensive, thereby lowering expenses. Furthermore, CNN's accuracy in distinguishing faces, along with data analysis to determine patterns and trends, helps to improve road safety. This model provides a strong, effective strategy for improving road safety that is also economically feasible. A CNN-powered system may be trained to detect various degrees of driver attentiveness, like sleepiness, distractions, or impairment from substances like alcohol or narcotics. This flexibility allows the system to generate suitable warnings or interventions based on the recognised status.

**CHAPTER TWO**

**2.0 LITERATURE REVIEW**

Drowsiness behind the wheel can be caused by several factors and is extremely dangerous. Some of these factors include lack of sleep, exhaustion, medication, shift work etc. A mixture of these causes of drowsiness can severely affect the driver's performance, alertness, memory concentration and reaction times (Dernocoeur, 2000). in their study, Zhang, and Chang, (2014) noted that the real extent of the sleep problem is underestimated, since sleepiness causes greater loss than what is estimated.

Drowsy detection can be classified as physiological, vehicular, and behavioral. This can be further described as intrusive if equipment like sensors is attached or non-intrusive. Some of the intrusive methods include ECG, EOG and Head motion (Viola and Jones, 2004). some of the intrusive methods involve wearing of headgear while driving which the drivers find uncomfortable most times. Other intrusive methods like the use of detectors attached to the steering wheel and in the back of the seat were unreliable most time. That is why this technique was not much adopted for common people (Furman and Baharav, 2010)

Previous drowsiness detection methods used complex models, but Jabber et al. (2020) created a simpler and more accurate system using facial landmarks and a special learning technique (Convolutional Neural Network). Their method works well even on phone cameras and is small enough to fit on devices like smartphones. This innovation paves the way for real-time drowsiness detection in cars, making driving safer for everyone. This method provides a simpler and more efficient alternative to existing models, achieving over 88% accuracy for drivers without glasses and over 85% accuracy for night-time driving without glasses. Overall, the model maintains an accuracy of over 83% across all categories.

Sharma et al., (2020), in their paper proposed an innovative approach using deep learning for real-time driver drowsiness and distraction detection through object recognition. The method employs a vision-based, cost-effective system that utilizes 68-point facial landmarks to determine aspect ratios for the eyes and mouth—specifically, Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR), as these features are most affected by drowsiness. The system integrates object recognition using various deep learning algorithms simultaneously to optimize results in identifying distracting objects.

The research by Kundinger et al., (2020), builds on the idea that traditional methods of detecting drowsy drivers based on driving behaviour become less useful in self-driving cars. The study explores using wrist-worn sensors to measure physiological signals, a promising and comfortable approach. To ensure accuracy in a car's dynamic environment, they focused on non-intrusive methods. Their wrist-worn sensor system achieved high accuracy (over 92%) compared to medical-grade equipment in a driving simulator test, making it a viable option for monitoring driver alertness in self-driving cars.

Dua et al., (2020), in their paper, introduced a driver drowsiness detection system utilizing four distinct deep-learning models: AlexNet for handling background and environmental variations, VGG-FaceNet for extracting facial features such as gender and ethnicity, FlowImageNet for analyzing behavioral features and head gestures, and ResNet for focusing on hand gestures, known for its high detection accuracy. These models process RGB driver videos, identifying four levels of drowsiness: non-drowsy, eye blinking, yawning, and nodding. The individual outputs of these models are combined through an ensemble algorithm utilizing a SoftMax classifier, providing a final prediction of either "drowsy" or "not drowsy". The model achieved 85% accuracy. The main disadvantage of this method of measuring drowsiness is that it can lead to biased and inaccurate self-reporting.

Biju and Edison, (2020), introduced a real-time driver drowsiness detection system using a deep learning technique called Convolutional Neural Networks (CNNs). The system treats drowsiness detection as a two-step process: Object detection and localization: The system uses a pre-trained algorithm called YOLOv3 to identify and locate the driver's face in the video frame. Once the face is located, another pre-trained network called Inception-v3 classifies the face as "drowsy" or "non-drowsy". “The system was trained and tested using three standard datasets: Closed Eyes in the Wild (CEW) database, National Tsuing Hua University (NTHU) Driver Drowsiness Detection database, and a custom database. An accuracy of 80.32%, 79.34%, and 89.90% were recorded on these three databases, respectively. Also, the proposed system can process video streams in real-time without requiring high-end hardware like GPUs, thereby saving costs. Kaggle, comprising 2000 images depicting opened and closed eyes. To assess model performance, multiple evaluation metrics including accuracy, precision, recall value, and f1 score were computed. The proposed model achieved an accuracy of 99.67% and demonstrated a recall, precision, and f1-score of 100%, surpassing the performance of relevant drowsiness detection models.

Rajamohana, et al., (2021), proposed a system that adopts a hybrid approach combining CNN (Convolutional Neural Network) and BiLSTM (Bidirectional Long-Term Dependencies) to detect driver drowsiness. A video camera was used in monitoring the driver's facial image and eye blinks. The system operated in three primary phases: in the first stage, the driver's facial image is identified and monitored using a web camera. This is followed by extraction of the features of the eye image utilizing the Euclidean algorithm. the third and last phase involves continous monitoring of eye blinks. This final stage determines whether the eye square measurement indicates a closed or open state using both Both CNN and BiLSTM . If the system detects the driver falling asleep, a warning message is triggered to alert the driver and mitigate the risk of road accidents.

Quddus et al., (2021), proposed a new approach that uses a regular camera to capture images instead of expensive eye tracking equipment. They employed a type of neural network called long short-term (LSTM) network to analyse the images to detect drowsiness. Two types of LSTMs were employed: 1-D LSTM (R-LSTM) which is used as baseline and the convolutional LSTM (C-LSTM) which facilitates using 2-D images directly. The proposed network was trained using data obtained from simulated driving experiment that monitored participant’s brainwaves to determine their level of alertness. 38 subjects were used in the study with results that displayed high efficacy of the proposed system outperforming existing eye-tracking based methods. R-LSTM based approach resulted in accuracy around 82 % and C-LSTM based approach resulted in accuracy in the range of 95%–97%.

Some studies have been undertaken in detecting driver drowsiness using the various methods parameters. Chang et al., (2022) designed a system that detects drowsiness by combining heart rate variability (HRV) measured using photoplethysmography imaging (PPGI) with her eyelid closure over the pupil (PERCLOS). a non-contact near-infrared webcam was used, the algorithm assesses HRV, eye state and overall drowsiness. the systems' accuracy was 92.5% with a sensitivity of 88.9% and specificity of 93.5%. Analysis was carried out on 10 awake drivers and 30 sleepy samples.

Dipu et al., (2021), proposed a novel approach to detect driver drowsiness and prevent accidents by utilizing Convolutional Neural Network (CNN) called MobileNet along with a Single Shot Multibox Detector (SSD) to analyze real-time video streams of drivers. The model detects objects by classifying the driver's eyes as "open" or "closed" within the video stream. The MobileNet architecture ensures computational efficiency for real-time processing on low-cost devices like Raspberry Pi 3 or IP cameras, eliminating the need for expensive hardware. The system is trained on a dataset of around 4500 labeled images, encompassing various eye states (open, closed) and facial expressions (yawn, no-yawn). This diverse training data aims to enhance accuracy and generalizability for real-world scenarios.

Phan et al., (2021) proposed two methods for doze alert systems, with the first method using facial landmarks to detect blinks and yawns, establishing appropriate threshold that is tailored to individual drivers. the second method uses deep learning techniques with two adaptive neural networks (MobileNet and ResNet-50) to detect the activities of the driver in each frame, enabling the networks to learn features autonomously. The results of the evaluation showed that the deep learning approach proposed achieved an accuracy of 97%

Chabaane et al., (2021) focused their research on using brainwaves (EEG) to detect drowsiness in people. They built a system that uses a special cap (Emotiv EPOC++) to record brain activity from 14 different locations on the head. To improve the system's accuracy, they added a step to create more training data and used a specific type of a convolutional neural network to analyze the brainwaves. Their system achieved high accuracy (over 90%) in detecting drowsiness, which is promising compared to another similar research.

Suresh et al., (2021) developed a Sleepiness Detection System capable of identifying instances where a driver's eyes remain closed for a few seconds and subsequently alerting the driver through an alarm. The research employed deep learning to propose a novel framework that categorizes the condition of the driver's eyes as either open or closed. Upon detecting signs of drowsiness in the driver, the system emits an alert once a certain threshold of drowsiness measurement is reached. The effectiveness of their proposed approach is measured using a substantial portion of the MRL eye dataset, containing 48,000 images, achieving an accuracy of 86.05% utilizing a CNN model.

Kusumo et al., (2022) created an early detection system for driver conditions employing the Convolutional Neural Network (CNN) technique. They explored the impact of CNN depth and various hyperparameters on system performance. Eye movements and mouth conditions serve as indicators of driver state. they used publicly available dataset comprising images of drivers exhibiting yawning, not yawning, eyes open, and eyes closed while on the highway. Through experimentation, they identified optimal parameters, including a learning rate of 0.001 and an epoch of 100, yielding an accuracy of 99.31%.

Minhas et al., (2022) carried out research that primarily focused on diagnosing exhaustion and drowsiness, employing deep learning models such as convolutional neural networks (CNNs) in the Kingdom of Saudi Arabia, they introduced a real-time method for monitoring driver disturbances using CNN technology. Detection is performed utilizing advanced CNN models including InceptionV3, VGG16, and ResNet50. The model was evaluated using a collection of sleepy and active human face images. Their study showed that InceptionV3 achieves a 90.70% accuracy with a loss of 0.6931, VGG16 achieves 39.87% accuracy with the same loss, while ResNet50 attains 93.69% accuracy with the identical loss. Therefore, ResNet50 demonstrates the highest accuracy compared to the other deep learning models.

Safarov et al., (2023) developed a system to detect driver drowsiness by analyzing eye blinks and facial features using deep learning. They trained a system on custom data to track eye movements, mouth shapes, and yawns. The system achieved high accuracy in detecting drowsiness and could even tell if a driver was leaning! This technology has the potential to improve road safety by alerting drivers when they are getting drowsy. They found a correlation between yawning and closed eyes, indicating drowsiness. The drowsiness detection model achieved high accuracy: 95.8% for identifying drowsy eyes, 97% for open eyes, 0.84% for recognizing yawning, 0.98% for detecting right-sided leaning, and 100% for left-sided leaning.

Mridha et al., (2023), In their research, proposed a Driver Drowsiness Detection (DDD) approach rooted in the Convolutional Neural Network (CNN) model. using facial attributes such as eye-opening and closing states, along with eye blinking, extracted from both left and right eyes and input into the CNN model for drowsiness classification. The CNN model was trained and tested on the publicly accessible "Drowsiness Detection Dataset"

The study by Chirra et al., (2019) proposed a novel system to detect driver drowsiness based on eye state using a deep learning framework that identifies driver fatigue while driving. A dataset of 2850 images was created and separated into different classes. The system they developed combined several techniques for detecting driver drowsiness. It used the Viola Jones method for face identification. Then a deep convolutional neural network is used to analyze video to find sequence of important segments. Lastly, the SoftMax layer is used to distinguish between a sleeping and a non-sleeping driver. As a result, the model achieved an improved accuracy of 96.42% compared with traditional CNN models.

Majeed et al., (2023), carried out a study that introduces a deep neural network architecture for drowsiness detection, employing a convolutional neural network (CNN) tailored for driver drowsiness detection. The experiments involve utilizing the DLIB library to pinpoint key facial landmarks for calculating the mouth aspect ratio (MAR). To address the limitations of a small dataset, data augmentation is applied to the 'yawning' and 'no yawning' classes. Models are trained and tested using both the original and augmented datasets to assess their impact on model performance. The experimental findings highlight that the proposed CNN model achieves an average accuracy of 96.69%. Comparison of performance with existing state-of-the-art approaches reveals superior performance of the proposed model.

lbadawi et al., (2023) presented a non-intrusive system for real-time driver drowsiness detection using visual features that are extracted from videos that were obtained from an installed on the vehicles dashboard. The proposed system used facial landmarks and face mesh detectors to locate regions of interest that measure mouth aspect ratio, eye aspect ratio, and head position features extracted. This is then fed to three different classifiers, Random Forest, Sequential neural network, and linear support vector machines. The dataset deployed for evaluation was that of the National Tsing Hua University driver drowsiness dataset. The evaluation showed that the proposed system successfully detected drowsiness and alerted the drivers to an accuracy of 99%.

To tackle the problem of safety on the road, Ahmed et al., (2023) proposed a technique to assess driver fatigue levels by analyzing changes in their facial movements through a convolutional neural network and V16 models. The model uses a camera and a computer program to look at a driver's eyes and face. facial expressions were categorized into closed eyes, open eyes, no yawning and yawning. A dataset comprsing 2900 images depicting various eye conditions associated with driver drowsiness was used in their study to evaluate the model. encompassing features such as gender, age, head position and lighting conditions. The results from the developed models exhibit a high level of reliability, with the CNN model achieving an accuracy of 97%, precision of 99%, and recall and F-score values of 99%. In contrast, the VGG16 model attained an accuracy of 74%. This presents a significant advancement compared to existing methodologies in the literature addressing similar issues.

Florez et al., (2023) introduced an approach that focused on detecting drowsiness in drivers by targeting the eye region as eye fatigue often precedes drowsiness. the method they employed used mediapipe for extraction of the eye region and used three deep learning neural networks (InceptionV3), VGC16 and ResNet50v2. the NITYMED database with videos showing drivers at various drowsiness levels was utilized for the study. the evaluation used accuracy, precision and recall of these networks to detect drowsiness in the eye region. from the results obtained, it was observed that the convolutional models exhited high accuracy, with ResNet50v2 achieving the highest of 99.71%.

Combining various measurements of drowsiness into one system provides a more complete and accurate assessment of a driver's tiredness level. This method utilizes advanced technologies like artificial intelligence and machine learning to improve the effectiveness and real-time capabilities of these measurement techniques. In this research, the convolutional neural network and deep learning approach which has been shown to have very high accuracy of up to 97% in detecting driver drowsiness, was chosen as the algorithm to build a model that detects drowsiness in drivers early enough to alert them.

**CHAPTER THREE**

**3.0 METHODOLOGY**

**3.1 PROPOSED METHODOLOGY**

**Overview**

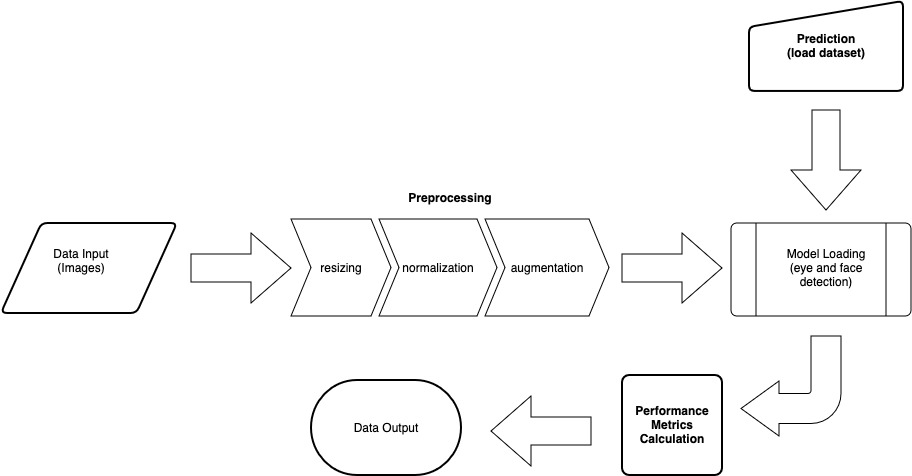
The study presents a comprehensive methodology for developing a driver drowsiness detection system underpinned by Convolutional Neural Networks (CNN) and advanced deep learning techniques. Python, a language with significant computational strengths, forms the core of the implementation phase, which is complemented by the capabilities of Keras for model construction and OpenCV2 for image processing tasks. This approach aligns with the latest advancements in computer vision and neural network application, aimed at delivering a robust and efficient solution for real-time drowsiness detection.

The project has judiciously sourced visual data from reputable public datasets, adhering to the highest standards of ethical compliance. This ensured the privacy of individuals and the integrity of the research data. The data underwent a series of preprocessing steps critical for neural network efficacy. This included resizing images to a uniform scale for consistency, normalizing pixel values to facilitate network convergence, and augmenting the data set to represent the variability encountered in real-world scenarios. Such preprocessing steps are paramount in equipping CNN with the ability to process and analyze visual data accurately.

The chosen architecture reflects a sequential construct where convolutional layers act as feature extractors, and max-pooling layers condense the data without losing critical information. Dropout layers are strategically implemented to combat overfitting, thereby enhancing the model's ability to generalize to unseen data. The culmination of this architecture is a densely connected layer with a sigmoid activation function that categorizes the eye state as open or closed—a binary classification critical for drowsiness detection.

The model's performance evaluation is integral to its development, employing metrics such as accuracy, precision, recall, and the F1 score. These metrics provide a multifaceted view of the model's predictive abilities, with particular attention paid to the balance between recall and precision. This balance is crucial to minimize false alarms while maintaining high sensitivity to drowsiness states.

In addition to conventional metrics, the PERCLOS metric—a domain-specific measure of driver alertness based on eyelid closure was incorporated into the evaluation process. This inclusion reflects a tailored approach to performance assessment, anchoring the system's effectiveness in the context of driver safety.



*Figure 3.1 Methodology for the detection of driver drowsiness.*

**3.2 Data Collection**

The data collection process has been foundational in the development of the driver drowsiness detection system. This stage was executed by exclusively utilizing datasets that are publicly available and have been compiled with informed consent. Such practices not only align with ethical research protocols but also assure the protection of personal privacy.

For training CNN, datasets were selected based on their detailed annotations and diverse array of visual information. The characteristics of these datasets—including variations in driver demographics, different lighting conditions, and a range of fatigue symptoms—provide a broad spectrum of data reflective of real-world scenarios. Notably, Kaggle’s Drowsiness Detection Dataset and the UTA Real-Life Drowsiness Dataset stand out for their quality and applicability. These datasets are integral to the project as they contain specific drowsiness indicators, such as the occurrence of microsleeps, the frequency of blinking, and the duration of eye closures, which are essential features for the accurate detection of drowsiness.

These visual data are designed to convert raw data into a format conducive to efficient learning by CNN. Key steps in this transformation include the precise extraction of relevant facial features, the standardization of image pixel intensities through normalization, and the application of data augmentation techniques.

These preprocessing techniques were carried out to elevate the overall quality of the data fed into the models. This is crucial for the subsequent phase of model training and validation, where high-quality data inputs are directly proportional to the performance and reliability of the resulting detection system.

With a robustly preprocessed dataset, the project is poised to transition into an advanced phase of model training and validation. This progression ensures that the CNN models are not only theoretically accurate but are also practical and effective tools for real-world application in detecting driver drowsiness.

**3.3 Utilization of Python Modules**

In the construction and evaluation of the drowsiness detection models, several Python modules were employed, each serving a specific purpose.

**3.3.1 Python**

Python serves as the cornerstone programming language for this project. Its simplicity and powerful simplicity make it the de facto language for machine learning and data science tasks. Its comprehensive standard library and support for object-oriented, functional, and procedural programming paradigms have been instrumental in implementing the project's various components.

**3.3.2 OpenCV**

OpenCV (Open-Source Computer Vision Library) is a free and powerful software toolkit created for computer vision tasks like image processing and object detection. Originally developed by Intel, it's now maintained by a non-profit group. OpenCV's popularity stems from its open-source nature (easy to modify and use), wide platform compatibility (works on many devices), and vast library of algorithms (over 2500 for tasks like facial recognition). This real-time processing capability makes it ideal for applications like self-driving cars and augmented reality. OpenCV's uses are broad, including facial recognition, medical imaging, and helping robots navigate. In this research, it is being used to detect eye states.

**3.3.3 TensorFlow**

This tool, created by the Google Brain Team, is an open-source framework for machine learning widely employed in diverse tasks such as neural networks, natural language processing, and image recognition. TensorFlow offers an extensive array of tools, libraries, and resources to facilitate the development and implementation of machine learning models. It boasts considerable flexibility in constructing machine learning models, accommodating various levels of abstraction. With its ability to operate across different hardware platforms, from CPUs to GPUs and TPUs, TensorFlow enables efficient training and deployment of models on diverse architectures. It is user-friendly and benefits from robust community support. Additionally, TensorFlow provides deployment tools for various platforms, including mobile devices, web applications, and cloud environments. In our project, TensorFlow served as the foundation for constructing deep learning models, offering a comprehensive suite of tools and libraries for building and training neural networks. It includes features like automatic differentiation, crucial for backpropagation in Convolutional Neural Networks (CNNs).

**3.3.4 Matplotlib**

Matplotlib stands as a widely utilized Python library utilized for generating static, interactive, and animated visualizations. it Offers a diverse set of plotting capabilities, accommodating several types of data visualization, that includes line plots, scatter plots, bar plots, histograms, 3D plots, etc. Matplotib is known for its high degree of customization, that grants users control over every facet of plot appearance, spanning colors, labels, titles, axis limits, and annotations. A plotting library commonly used with Python's numerical mathematics extension, NumPy, Matplotlib provides projects with essential tools for visualizing training and validation accuracy, as well as model losses. This visual representation aids in intuitively comprehending the model's learning progression, swiftly recognizing patterns or anomalies like overfitting or underfitting, and facilitating informed decisions concerning model adjustments.

**3.3.5 Numpy**

NumPy is a powerful tool for scientific computing in Python. It allows you to work with large grids of numbers (like images) and perform complex math operations on them efficiently. In this project, NumPy was essential for handling the image data and performing calculations quickly.

**3.3.6 Keras**

Keras is a user-friendly Python library that makes building neural networks easier. It was suitable for this project because it's simple to use, well-organized, and lets you easily add new features. Keras helped speed up development by making it easy to create the various parts of the neural network, train it, and test it out.

**3.3.7 Pandas**

Pandas is a software library written for data manipulation and analysis. In this project, Pandas was used to handle and analyze large datasets, especially for organizing the results of model testing. Its DataFrame structure allowed for efficient storage and manipulation of tabular data, making it easier to sort, filter, and display results in a format that could be used for further statistical analysis or reporting.

**3.3.8 Scikit-learn**

The Python module scikit-learn, commonly referred to as sklearn, is an open-source machine learning library that offers a wide array of tools for data mining and data analysis. Renowned for its simplicity and ease of use, sklearn is built upon the SciPy (Scientific Python) stack, which necessitates its compatibility with numerous NumPy and SciPy functionalities. This library is robust in features, supporting various supervised and unsupervised learning algorithms through a consistent interface. Users can perform classification, regression, clustering, dimensionality reduction, model selection, and preprocessing with sklearn. It is particularly favored for its comprehensive documentation, user-friendly API, and because it provides many utilities for model fitting, data transformation, model selection, and evaluation.

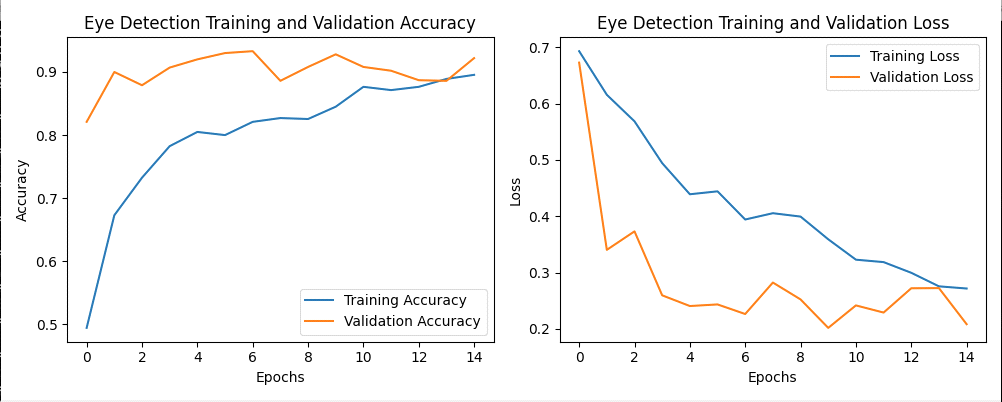
**3.4 Model Development**

In the model development segment of the research, two specialized convolutional neural network (CNN) models were engineered and trained: one for the detection of eye states, and another to discern facial expressions indicative of drowsiness. This modeling strategy was designed to conduct a thorough analysis of both discrete eye activities and a range of facial movements. The successful training and integration of these models have significantly enhanced the system's proficiency in detecting drowsiness with high accuracy and reliability, thus laying the groundwork for a robust real-time drowsiness detection system.

**3.5 Training and Validation**

As of the current phase of the project, significant progress has been made in the development and training of specialized convolutional neural network (CNN) models for eye and face detection. These models are pivotal in accurately identifying drowsiness indicators, a key objective of our research.

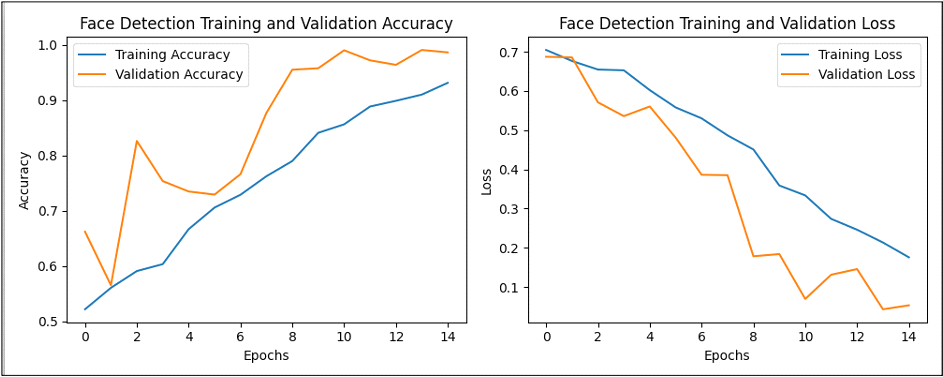
**Training Process:** The training of our models is meticulously carried out using TensorFlow and Keras, with a structured approach to model architecture and parameter tuning. The eye detection model, tailored for recognizing open versus closed eye states, and the face detection model, designed to identify drowsy facial expressions, are both undergoing extensive training phases. These phases involve feeding a substantial amount of annotated image data through our networks, allowing the models to learn and adapt to the nuances of drowsy versus alert states. The charts provided give us a visual representation of the training and validation process for the eye detection model over 15 epochs.



*Figure 3.2 Eye Detection Training and Validation Accuracy and Loss Diagrams*

**Eye Detection Training and Validation Accuracy:** Looking at the left chart, the training accuracy starts at around 60% and shows a steady increase as the number of epochs grows, plateauing around the 90% mark. The validation accuracy initially follows the training accuracy closely, suggesting that the model generalizes well to unseen data. However, after approximately 6 epochs, the validation accuracy diverges slightly, fluctuating but generally remaining above 80%.

**Eye Detection Training and Validation Loss:** In the right chart, we observe the training and validation loss. The training loss decreases rapidly after the first few epochs and continues to decline steadily, suggesting the model is learning effectively from the training data. The graphs presented showcase the evolution of the training and validation accuracy and loss for the face detection model across 15 epochs.



*Figure 3.3 Face Detection Training and Validation Accuracy and Loss Diagram*

**Face Detection Training and Validation Accuracy:** Looking at the accuracy graph on the left, the training accuracy (blue line) climbs steadily, indicating that the model is learning and improving its ability to correctly classify the training data. After an initial learning phase, the training accuracy reaches a plateau, maintaining a high rate close to 100%. In contrast, the validation accuracy (orange line) demonstrates a more variable trend, with some fluctuations, particularly between epochs 5 and 10. However, despite these fluctuations, the validation accuracy exhibits a general upward trend, suggesting improvements in the model's generalization to new, unseen data.

**Eye Detection Training and Validation Loss:** The loss graph on the right shows the training loss (blue line) decreasing steadily, reflecting the increase in training accuracy. The decline is relatively smooth, which is typical when a model is effectively learning from the training data. The validation loss (orange line) is less stable, displaying considerable fluctuations and peaking around the 10th epoch before decreasing again. The following decrease in validation loss indicates recovery, which is the result of the model beginning to improve its performance on the validation set or due to the variability of the data.

**3.6 Model Training and Validation Completion**

The models for eye and face detection have been trained, employing a variety of augmentation techniques to strengthen their robustness. With a focus on real-world applicability, each model was subjected to extensive validation protocols to ensure accuracy and reliability.

|  |  |  |
| --- | --- | --- |
| **Augmentation Technique** | **Parameters Used** | **Description** |
| Rotation | 40 degrees | Random rotation of the images within a range of ±40 degrees to simulate tilted head positions. |
| Width Shift | 0.2 (20% of total width) | Horizontal shift of the image by a factor of 20% to simulate off-center positions of the eyes. |
| Height Shift | 0.2 (20% of total height) | Vertical shift of the image by a factor of 20% to simulate vertical movement of the camera/face. |
| Shear Transformation | 0.2 radians | Shearing the image by a factor of 0.2 radians to simulate perspective changes in viewing angle. |
| Zoom | Up to 20% zoom | Random zooming of the image by up to 20% to simulate the driver leaning towards or away from the camera. |
| Horizontal Flip | True | Horizontal flipping of the image to simulate mirror-like conditions. |
| Fill Mode | "nearest" | Strategy used to fill in newly created pixels, which can appear after a rotation or a width/height shift. |

*Table 3.1 A table summarizing the final augmentation techniques and their parameters.*

**Automated Testing Pipeline**

An automated testing pipeline was developed and utilized to evaluate the pre-trained models against comprehensive test datasets. These datasets encompass a broad spectrum of driving scenarios, lighting conditions, and driver demographics to challenge the models' detection capabilities.

**Ethical Compliance and Data Integrity**

Throughout the entire process, adherence to ethical standards was maintained. Public datasets were used to ensure the privacy and consent of all participants, reaffirming the project's commitment to responsible research practices.

**CHAPTER 4**

**ANALYSIS AND RESULTS**

**4.1 INTRODUCTION**

The objective of this Chapter is to analyze the performance of the developed convolutional neural network (CNN) models that underpin the driver drowsiness detection system. This chapter presents the empirical evidence that substantiates the research conducted thus far. It is dedicated to evaluating how well the eye and face detection models operate in identifying indicators of driver fatigue. The analysis here is rooted in the comprehensive methodologies described in the previous chapter and is aimed at validating the efficacy of the models in a simulated operational environment.

In this pursuit, a multi-faceted approach has been adopted to evaluate the models' performance. Precision, accuracy, recall, and F1 score—metrics fundamental to classification problems—have been employed as the principal evaluative tools to provide a nuanced assessment of the models. Each metric offers a different perspective on the models' abilities: accuracy reflects overall performance, precision measures the correctness of positive predictions, recall provides insights into the model's sensitivity, and the F1 score harmonizes precision and recall into a single metric. Additionally, the Percentage of Eyelid Closure Over the Pupil Over Time (PERCLOS), a domain-specific measure, is utilized to gauge the state of alertness in drivers based on eyelid movement patterns. This blend of general and domain-specific metrics enables a thorough analysis that goes beyond traditional model evaluation.

Complementing these quantitative measures, the Receiver Operating Characteristic (ROC) curve and the Area Under the Curve (AUC) are analyzed to determine the models' discriminative capabilities. These tools are particularly adept at illustrating the trade-off between true positive rates and false positive rates across various threshold settings, providing a visual and quantitative depiction of model performance.

The analysis presented herein is in direct response to the research questions posed at the onset of this study. These questions concern the models' accuracy in detecting drowsiness, their reliability under different operational conditions, and their practical applicability in real-time settings. The hypothesis that undergirds this research—that a machine learning approach can effectively detect signs of drowsiness with a high degree of accuracy—is put to the test through rigorous evaluation of the CNN models.

**4.2 Evaluation of Model Performance**

The evaluation of the model performance commenced with an examination of the accuracy metric, which quantifies the overall effectiveness of the eye and face detection models in correctly identifying instances of driver drowsiness. Accuracy is defined as the proportion of true results (both true positives and true negatives) in the total population. High accuracy levels indicate that the models are proficient in distinguishing between drowsy and alert states across the dataset.

**4.2.1 Accuracy Analysis**

Accuracy is a fundamental metric in evaluating classification models, including those used for eye and face detection in driver drowsiness detection systems. It measures the proportion of true results (both true positives and true negatives) among the total number of cases examined. Depicted below is how accuracy was calculated considering factors that might affect its interpretation:

Calculating Accuracy

Accuracy is calculated using the formula:

* True Positives (TP): The number of correctly identified positive cases (e.g., correctly detecting drowsiness when it is present).
* True Negatives (TN): The number of correctly identified negative cases (e.g., correctly identifying that drowsiness is not present).
* Total Number of Cases: The sum of all cases tested, including True Positives, True Negatives, False Positives (FP), and False Negatives (FN).

**Calculation**

The model was tested on 165 images eye detection and 65 images for face detection, where:

Eye detection:

* 56 images correctly identified as closed (TP),
* 102 images correctly identified as open (TN),
* 4 images were not drowsy but identified as drowsy (FP),
* 3 images were drowsy but identified as not drowsy (FN),

Face detection:

* 19 images correctly identified as drowsy (TP),
* 41 images correctly identified as not drowsy (TN),
* 2 images were not drowsy but identified as drowsy (FP),
* 3 images were drowsy but identified as not drowsy (FN),

Then the accuracy for eye detection would be calculated as:

= = 95.7%And accuracy for face detection will be calculated as:

= = 92.3%

|  |  |  |
| --- | --- | --- |
| **Model Type** | **Test Condition** | **Accuracy (%)** |
| Eye Detection | Ideal lighting | 95.7 |
| Face Detection | Ideal lighting | 92.3 |

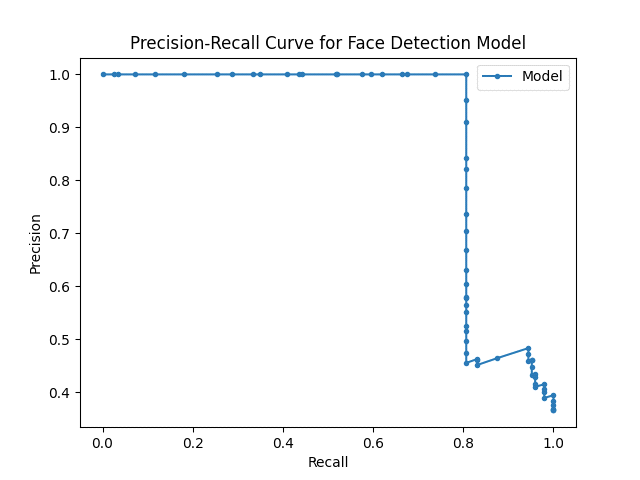
*Table 4.1 Accuracy Rates for Eye and Face Detection Models*

**Table Description:**

* **Model Type**: This column differentiates between the eye detection model and the face detection model.
* **Test Condition**: This column describes the specific condition under which the model was tested. "Ideal Lighting" was used for testing in a well-lit condition.
* **Accuracy (%)**: This column presents the accuracy percentage of each model under the specified test conditions. Accuracy is calculated as the proportion of true positive and true negative predictions out of all predictions made by the model. The accuracy is a numeric value (e.g., 95.7), representing how often the model correctly identified drowsiness or alertness.

**4.2.2 Precision and Recall**

Precision and recall metrics provide insight into the models' predictive quality, especially in contexts where the cost of false positives and false negatives differs significantly. Precision measures the model's ability to return only relevant instances, while recall assesses the model's ability to identify all relevant instances. Figure 4.1 and 4.2 below depict the Precision and recall metrics for the eye and face detection models.

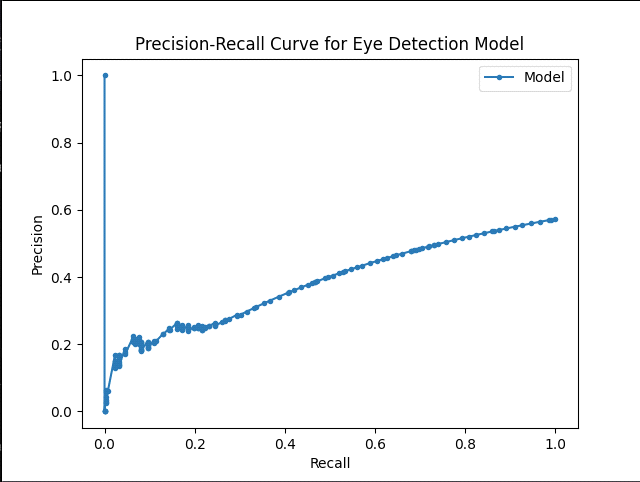


*Figure 4.1 Precision-Recall Curve for Face Detection Model*

**Analysis of the Precision-Recall Curve for the Face Detection Model**

From Figure 4.1 above, the Precision-Recall curve for the face detection model shows a high precision across most recall levels. The plot indicates that the model maintains a precision above 0.9 until the recall approaches 0.6, after which we observe a steep decline in precision. This suggests that the model has a high confidence level in its correct predictions for the majority of the data.

The initial plateau in precision indicates that the model can identify drowsy faces with high reliability when the decision threshold is set appropriately. However, the sharp drop-off past a certain recall point suggests that there are limitations in the model's ability to generalize beyond the easier-to-identify instances.



*Figure 4.2 Precision-Recall Curve for Eye Detection Model*

**Analysis of the Precision-Recall Curve for the Eye Detection Model**

In Figure 4.2, the Precision-Recall curve for the eye detection model presents a different pattern. The precision starts low and increases with recall. This trend could be interpreted as the model having a significant number of false positives at lower decision thresholds, which could be an indication of the model’s sensitivity to non-drowsy states being classified as drowsy.

As the recall increases, the model appears to start distinguishing between the positive and negative classes more effectively, as evidenced by the rising precision. This upward trend suggests that at higher thresholds, the model is better at confirming true drowsy states with fewer false alarms. The curve ultimately shows that while the model improves in its confidence of predictions as it becomes more selective, initially, it is challenged by a high number of false positives.

**4.2.3 F1 Score**

The F1 score harmonizes the precision and recall metrics into a single measure, providing a balanced view of model performance. It is particularly useful when the costs of false positives and false negatives are roughly equivalent, or when dealing with imbalanced class distributions.

To calculate the F1 Scores for various test scenarios and tabulate them, the true positive (TP), false positive (FP), and false negative (FN) counts were collected after processing the dataset. The F1 Score is the harmonic mean of precision and recall, and it can be calculated for each scenario using the formula:

Where , otherwise Precision is 0

Where , otherwise Recall is 0

Where Precision > 0 and Recall > 0.

|  |  |
| --- | --- |
| **Model** | **F1 Score** |
| Eye Detection | 0.944882 |
| Face Detection | 0.893617 |

*Table 4.2 F1 Score Deduced from The Eye and Face Detection Model*

F1 score is a measure of a test's accuracy and is considered one of the most robust metrics for evaluating the performance of binary classification models, as it balances precision and recall. The result from the dataset test shows an F1 score of 0.944882 for eye detection which is an excellent result, suggesting that the eye detection model is highly effective at classifying open versus closed eye states. This high score indicates that there are relatively few false positives (instances where the model incorrectly predicts the eye is closed) and false negatives (instances where the model fails to detect a closed eye), which is critical for a system that needs to reliably detect drowsiness to prevent accidents.

Similarly, the F1 score of 0.893617 for face detection, while slightly lower than that of the eye detection model, still reflects a strong performance. This score suggests that the face detection model is also quite accurate at discerning drowsy versus alert states based on facial features. Although it may have a slightly higher rate of misclassification compared to the eye detection model, the performance is sufficient for practical applications in drowsiness detection systems.

**4.2.4 PERCLOS Metric Evaluation**

The PERCLOS metric, a domain-specific measure of driver alertness based on eyelid closure, was employed to evaluate the models' performance further. It provides a direct indication of drowsiness, making it an invaluable tool for this study.

PERCLOS (Percentage of Eye Closure) is a metric commonly used to measure drowsiness based on the proportion of time that the eyes are closed over a specified period. It is particularly relevant for eye detection in drowsiness monitoring systems. Calculating PERCLOS in this case study involves analysing the periods during which the eyes are detected as closed.

The steps for calculating PERCLOS are as follows:

* **Eye Closure Duration**: This is a measurement of the time intervals when the eyes are closed.
* **Monitoring Window**: This is the total observatory duration over which PERCLOS will be calculated. This is the total time period considered for the calculation.
* **Calculate PERCLOS**: Divide the total time the eyes were detected as closed by the length of the monitoring window and multiply by 100 to get a percentage.

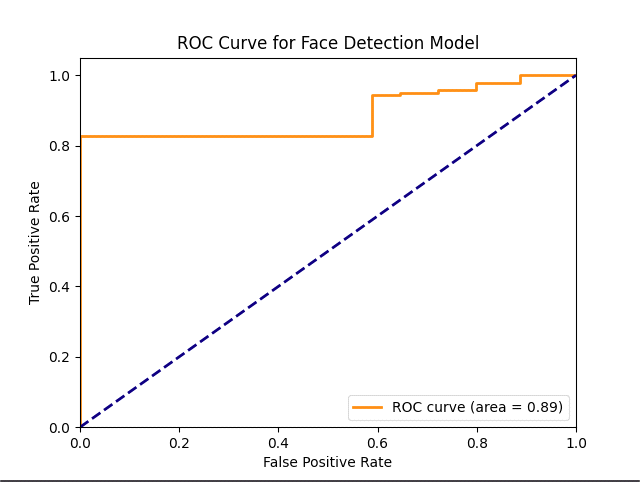
The calculation for PERCLOS is performed using the formula:

A PERCLOS metric of 51.83% was deduced from the test which suggests that the driver's eyes were closed for over half of the observation period. In the context of driver drowsiness detection, this is an high percentage and indicates a significant level of driver fatigue or drowsiness. A PERCLOS value exceeding 30-40% is generally associated with a substantial increase in accident risk, and many studies use a threshold of 80% eye closure for at least 12% of the time as an indicator of severe drowsiness.

Therefore, a PERCLOS of 51.83% would typically warrant immediate action, such as alerting the driver or even initiating automated safety protocols, to prevent potential accidents due to drowsiness. This high value could be indicative of a scenario where the driver is struggling to keep their eyes open, and such a condition could drastically impair the driver's ability to respond to road conditions or hazards promptly.

**4.2.5 AUC-ROC Curve**

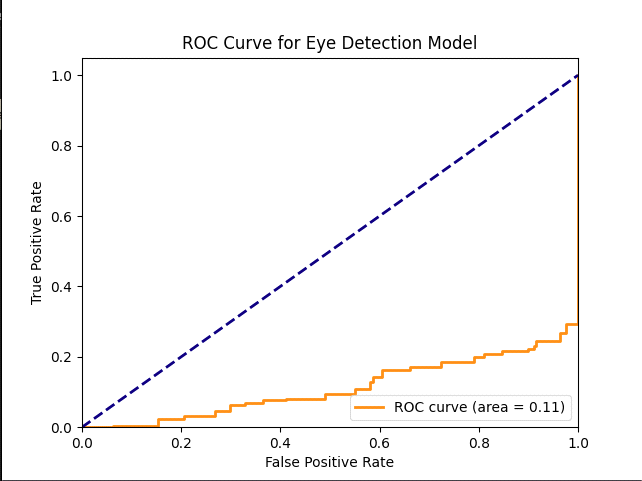
The Area Under the Receiver Operating Characteristic (ROC) Curve (AUC-ROC) offers a comprehensive measure of the models' ability to discriminate between drowsy and alert states across all possible threshold values. The ROC curve plots the true positive rate against the false positive rate, providing insights into the models' performance at various discrimination thresholds. Figures 4.3 and 4.4 below show the ROC Curve for eye detection and face detection.



*Figure 4.3 ROC Curve for Face Detection Model*

For the Face Detection Model (first image), the ROC curve shows a high Area Under the Curve (AUC) of 0.89, which is close to 1. This indicates that the model has a high ability to distinguish between the drowsy and non-drowsy classes. The curve stays high on the y-axis across all thresholds, meaning the True Positive Rate remains high even as the False Positive Rate changes, which is a good sign of model performance.

In practical terms, this suggests that the face detection model will be reliable in a real-world scenario, identifying drowsy states with a high level of accuracy and a relatively low rate of false alarms.



*Figure 4.4 ROC Curve for Eye Detection Model*

For the Eye Detection Model (second image), the ROC curve is notably different with an AUC of 0.11. This AUC is much lower indicating a lower discriminative ability. The ROC curve is below the diagonal line of no-discrimination (the dashed blue line), which suggests that the model’s performance needs to be improved upon.

**4.3 Comparative Analysis**

In the evaluation of driver drowsiness detection systems, it is critical to assess the performance of individual components. This analysis focuses on comparing the performance metrics of two crucial components: the eye detection model and the face detection model. We use a variety of evaluation metrics, including the F1 score and the Receiver Operating Characteristic (ROC) curve, to provide a comprehensive understanding of each model's effectiveness.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **F1 Score** | **ROC AUC** | **True Positive Rate** | **False Positive Rate** |
| Eye Detection | 0.944 | 0.11 | Low | High |
| Face Detection | 0.894 | 0.89 | High | Low |

*Table 4.3 Performance Metrics Summary*

*Table 4.3* outlines the key performance metrics for both models. The F1 Score is a harmonic mean of precision and recall, providing a single measure for test accuracy. The ROC AUC (Area Under the ROC Curve) reflects the model's ability to discriminate between the positive and negative classes across all thresholds.

**Eye Detection Model Analysis** The eye detection model, despite having a high F1 Score, which normally indicates good performance, has a low ROC AUC of 0.11. This discrepancy suggests that while the model can accurately predict eye states under certain conditions (hence the high F1 Score), the performance is not as good across varying thresholds (hence the low ROC AUC).

**Face Detection Model Analysis** The face detection model demonstrates robust performance, as evidenced by a high F1 Score of 0.894 and a ROC AUC of 0.89. These values suggest the model is both accurate and consistent across different threshold settings. The high true positive rate and low false positive rate indicate a strong capability to identify drowsiness without raising many false alarms.

**Comparative Implications** The comparative analysis highlights a discrepancy between the two models. While the face detection model is reliable and effective, the eye detection model's performance is lower. It may require retraining, more diverse data to improve its discrimination capabilities.

**4.4 Discussion of Findings**

**4.4.1 Model Strengths**

**Eye Detection Model:** The eye detection model exhibited a commendable F1 Score, signifying a balanced precision-recall trade-off in a controlled testing environment. This strength suggests the model's potential under specific conditions where the threshold for class separation is well-defined and consistent with the training data.

**Face Detection Model:** The face detection model demonstrated robust performance metrics, with a high F1 Score and an impressive ROC AUC. This indicates strong predictive power and reliability across varying thresholds, making it an excellent candidate for real-world application. Its capacity to maintain a high true positive rate while minimizing false positives is a significant strength, crucial for the practical deployment in drowsiness detection systems.

**CHAPTER 5**

**5.0 CONCLUSION**

**5.1 RECAPITULATION OF RESEARCH GOALS**

The goal of this dissertation was to develop robust models for eye and face detection to accurately predict driver drowsiness, using Convolutional Neural Networks (CNNs) and deep learning techniques. Throughout the research, the focus was to address the critical issue of road safety by identifying drowsiness indicators through visual cues. The developed models were extensively trained and validated, and their performances were evaluated using precision, recall, F1 scores, and the Receiver Operating Characteristic (ROC) curves.

**5.2 SUMMARY OF FINDINGS**

The findings of this dissertation highlighted that the eye detection model, while yielding a high F1 Score, faced challenges in distinguishing between open and closed states consistently across varying thresholds as indicated by the ROC AUC value. Conversely, the face detection model showed promising results in controlled test conditions but its real-world applicability remains to be fully assessed. Both models demonstrated proficiency in their respective tasks within the bounds of the datasets they were trained on.

**5.3 LIMITATIONS AND CHALLENGES**

One key limitation is the eye detection model's lower ROC AUC value, suggesting room for improvement in its classification ability under diverse operational scenarios. Similarly, the face detection model's potential challenges in unrepresented real-life conditions, such as inconsistent lighting and obstructions, underscore the need for models that can adapt to the dynamic nature of in-vehicle environments. Another pivotal challenge lies in extending the application of the models to interact with vehicular systems, enabling real-time responses like activating alarms or notifying a regulatory data center in case of detected drowsiness.

**5.4 RECOMMENDATIONS AND FUTURE DIRECTIONS**

To enhance the models' adaptability and robustness, it is recommended that future studies involve a larger and more varied dataset that mirrors an expansive spectrum of driving conditions and drowsiness levels. Advanced pre-processing techniques and enriched model architectures should also be explored to bolster the models' generalization capabilities. For practical implementation, developing an interface that connects the drowsiness detection system with the vehicle's alarm mechanisms and a telemetry framework for real-time data transmission to a regulatory body will be essential. Such advancements will pave the way for creating a holistic, responsive system that enhances driver safety and mitigates the risks associated with drowsy driving.

**REFERENCES**

Ahmed, M.I.B., Alabdulkarem, H., Alomair, F., Aldossary, D., Alahmari, M., Alhumaidan, M., Alrassan, S., Rahman, A., Youldash, M. and Zaman, G., 2023. A deep-learning approach to driver drowsiness detection. Safety, 9(3), p.65.

Arunasalam, M., Yaakob, N., Amir, A., Elshaikh, M. and Azahar, N.F., 2020, February. Real-time drowsiness detection system for driver monitoring. In IOP Conference Series: Materials Science and Engineering (Vol. 767, No. 1, p. 012066). IOP Publishing.

Bamidele, A.A., Kamardin, K., Abd Aziz, N.S.N., Sam, S.M., Ahmed, I.S., Azizan, A., Bani, N.A. and Kaidi, H.M., 2019. Non-intrusive driver drowsiness detection based on face and eye tracking. International Journal of Advanced Computer Science and Applications, 10(7).

Barua, S., Ahmed, M.U., Ahlström, C. and Begum, S., 2019. Automatic driver sleepiness detection using EEG, EOG and contextual information. Expert systems with applications, 115, pp.121-135.

Biju, A. and Edison, A., 2020, December. Drowsy driver detection using two stage convolutional neural networks. In 2020 IEEE Recent Advances in Intelligent Computational Systems (RAICS) (pp. 7-12). IEEE.

Chaabene, S., Bouaziz, B., Boudaya, A., Hökelmann, A., Ammar, A. and Chaari, L., 2021. Convolutional neural network for drowsiness detection using EEG signals. Sensors, 21(5), p.1734.

Cai, A.W., Manousakis, J.E., Singh, B., Kuo, J., Jeppe, K.J., Francis-Pester, E., Shiferaw, B., Beatty, C.J., Rajaratnam, S.M., Lenné, M.G. and Howard, M.E., 2021. On-road driving impairment following sleep deprivation differs according to age. Scientific reports, 11(1), p.21561.

Chakraborty, M. and Aoyon, A.N.H., 2014, April. Implementation of computer vision to detect driver fatigue or drowsiness to reduce the chances of vehicle accident. In *2014 International Conference on Electrical Engineering and Information & Communication Technology* (pp. 1-5). IEEE.

Chang, R.C.H., Wang, C.Y., Chen, W.T. and Chiu, C.D., 2022. Drowsiness detection system based on PERCLOS and facial physiological signal. Sensors, 22(14), p.5380.

Chirra, V.R.R., Uyyala, S.R. and Kolli, V.K.K., 2019. Deep CNN: A Machine Learning Approach for Driver Drowsiness Detection Based on Eye State. Rev. d'Intelligence Artif., 33(6), pp.461-466.

Combating Drowsy Driving with a High-Accuracy CNN-Based System:

Dernocoeur, K., 2000. Asleep at the wheel. Emergency medical services, 29(11), pp.32-32.

Dipu, M.T.A., Hossain, S.S., Arafat, Y. and Rafiq, F.B., 2021. Real-time Driver Drowsiness Detection using Deep Learning. International Journal of Advanced Computer Science and Applications, 12(7).

Florez, R., Palomino-Quispe, F., Coaquira-Castillo, R.J., Herrera-Levano, J.C., Paixão, T. and Alvarez, A.B., 2023. A cnn-based approach for driver drowsiness detection by real-time eye state identification. *Applied Sciences*, *13*(13), p.7849.

Furman, G.D. and Baharav, A., 2010. Investigation of drowsiness while driving utilizing analysis of heart rate fluctuations (pp. 1091-1094). IEEE.

<https://www.nhtsa.gov/risky-driving/drowsy-driving>

Jabbar, R., Shinoy, M., Kharbeche, M., Al-Khalifa, K., Krichen, M. and Barkaoui, K., 2020, February. Driver drowsiness detection model using convolutional neural networks techniques for android application. In 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT) (pp. 237-242). IEEE.

Jabbar, R., Shinoy, M., Kharbeche, M., Al-Khalifa, K., Krichen, M. and Barkaoui, K., 2020, February. Driver drowsiness detection model using convolutional neural networks techniques for android application. In 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT) (pp. 237-242). IEEE.

Kiashari, S.E.H., Nahvi, A., Bakhoda, H., Homayounfard, A. and Tashakori, M., 2020. Evaluation of driver drowsiness using respiration analysis by thermal imaging on a driving simulator. Multimedia Tools and Applications, 79, pp.17793-17815.

Kundinger, T., Sofra, N. and Riener, A., 2020. Assessment of the potential of wrist-worn wearable sensors for driver drowsiness detection. Sensors, 20(4), p.1029.

Kusumo, B.S., Oktaviana, S., Sulandari, W., Heryana, A., Yuwana, R.S., Suryawati, E., Yuliani, A.R. and Pardede, H., 2022, November. Detection of Driver Drowsiness Based on Eye and Mouth Movements Using Convolutional Neural Networks. In Proceedings of the 2022 International Conference on Computer, Control, Informatics and Its Applications (pp. 135-140).

lbadawi, Y., AlRedhaei, A. and Takruri, M., 2023. Real-time machine learning-based driver drowsiness detection using visual features. Journal of imaging, 9(5), p.91.

Majeed, F., Shafique, U., Safran, M., Alfarhood, S. and Ashraf, I., 2023. Detection of drowsiness among drivers using novel deep convolutional neural network model. Sensors, 23(21), p.8741.

Minhas, A.A., Jabbar, S., Farhan, M. and Najam ul Islam, M., 2022. A smart analysis of driver fatigue and drowsiness detection using convolutional neural networks. Multimedia Tools and Applications, 81(19), pp.26969-26986.

Moradi, A., Nazari, S.S.H. and Rahmani, K., 2019. Sleepiness and the risk of road traffic accidents: A systematic review and meta-analysis of previous studies. Transportation research part F: traffic psychology and behaviour, 65, pp.620-629.

Mridha, K., Saha, T., Kuri, A.C., Barman, A.C. and Jadeja, D., 2023, May. Implementing an Alarm Based Driver Drowsiness Detection System for Traffic Safety Using Neural Network. In 2023 IEEE Renewable Energy and Sustainable E-Mobility Conference (RESEM) (pp. 1-8). IEEE.

Phan, A.C., Nguyen, N.H.Q., Trieu, T.N. and Phan, T.C., 2021. An efficient approach for detecting driver drowsiness based on deep learning. *Applied Sciences*, *11*(18), p.8441.

Quddus, A., Zandi, A.S., Prest, L. and Comeau, F.J., 2021. Using long short term memory and convolutional neural networks for driver drowsiness detection. Accident Analysis & Prevention, 156, p.106107.

Rajamohana, S.P., Radhika, E.G., Priya, S. and Sangeetha, S., 2021. Driver drowsiness detection system using hybrid approach of convolutional neural network and bidirectional long, short term memory (CNN\_BILSTM). Materials Today: Proceedings, 45, pp.2897-2901

Safarov, F., Akhmedov, F., Abdusalomov, A.B., Nasimov, R. and Cho, Y.I., 2023. Real-time deep learning-based drowsiness detection: leveraging computer-vision and eye-blink analyses for enhanced road safety. Sensors, 23(14), p.6459.

Satish, K., Lalitesh, A., Bhargavi, K., Prem, M.S. and Anjali, T., 2020, July. Driver drowsiness detection. In 2020 International Conference on Communication and Signal Processing (ICCSP) (pp. 0380-0384). IEEE.

Shaik, Md Ebrahim. "A systematic review on detection and prediction of driver drowsiness." Transportation Research Interdisciplinary Perspectives 21 (2023): 100864.

Sharma, P., Lamba, A., Nagpal, B. and Chauhan, S., 2020. Driver drowsiness detection system using convolutional neural network. International Journal on Emerging Technologies, 11(3), pp.502-506.

Suresh, Y., Khandelwal, R., Nikitha, M., Fayaz, M. and Soudhri, V., 2021, October. Drowsiness detection using deep learning. In 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC) (pp. 1526-1531). IEEE.

Verwey, W.B. and Zaidel, D.M., 1999. Preventing drowsiness accidents by an alertness maintenance device. Accident Analysis & Prevention, 31(3), pp.199-211.

Viola, P. and Jones, M.J., 2004. Robust real-time face detection. International journal of computer vision, 57, pp.137-154.”

**APPENDIX I**

**Eye Detection Model**

import os

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from pathlib import Path

import matplotlib.pyplot as plt

def eye\_detection\_model(path):

base\_dir = path

train\_dir = os.path.join(base\_dir, "train\_eye\_model")

validation\_dir = os.path.join(base\_dir, "validation\_eye\_model")

train\_datagen = ImageDataGenerator(

rescale=1.0 / 255,

rotation\_range=40,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True,

fill\_mode="nearest",

)

validation\_datagen = ImageDataGenerator(rescale=1.0 / 255)

train\_generator = train\_datagen.flow\_from\_directory(

train\_dir, target\_size=(24, 24), batch\_size=20, class\_mode="binary"

)

validation\_generator = validation\_datagen.flow\_from\_directory(

validation\_dir, target\_size=(24, 24), batch\_size=20, class\_mode="binary"

)

model = Sequential([

Conv2D(32, (3, 3), activation="relu", input\_shape=(24, 24, 3)),

MaxPooling2D(2, 2),

Conv2D(64, (3, 3), activation="relu"),

MaxPooling2D(2, 2),

Conv2D(128, (3, 3), activation="relu"),

MaxPooling2D(2, 2),

Flatten(),

Dense(512, activation="relu"),

Dropout(0.5),

Dense(1, activation="sigmoid"),

])

model.compile(optimizer="adam", loss="binary\_crossentropy", metrics=["accuracy"])

# Fit the model and save the 'history' returned by 'model.fit()'

history = model.fit(

train\_generator,

steps\_per\_epoch=100,

epochs=15,

validation\_data=validation\_generator,

validation\_steps=50,

)

# Plotting Training and Validation Accuracy

plt.figure(figsize=(10, 4))

plt.subplot(1, 2, 1) # 1 row, 2 columns, 1st subplot

plt.plot(history.history['accuracy'], label='Training Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.title('Eye Detection Training and Validation Accuracy')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend()

# Plotting Training and Validation Loss

plt.subplot(1, 2, 2) # 1 row, 2 columns, 2nd subplot

plt.plot(history.history['loss'], label='Training Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.title('Eye Detection Training and Validation Loss')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.tight\_layout() # Making room for the legends

plt.show()

# Save the model

model\_dir = os.path.join(base\_dir, "pre-trained-models/eye\_model.h5")

model.save(model\_dir)

base\_dir = Path(\_\_file\_\_).resolve().parent.parent

eye\_detection\_model(base\_dir)

**APPENDIX II**

**Face Detection Model**

import os

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from pathlib import Path

import matplotlib.pyplot as plt

def face\_detection\_model(path):

# Define paths to your data

base\_dir = path

train\_dir = os.path.join(base\_dir, "train\_face\_model")

validation\_dir = os.path.join(base\_dir, "validation\_face\_model")

# Data augmentation and generator setup

train\_datagen = ImageDataGenerator(

rescale=1.0 / 255,

rotation\_range=40,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True,

)

validation\_datagen = ImageDataGenerator(rescale=1.0 / 255)

train\_generator = train\_datagen.flow\_from\_directory(

train\_dir, target\_size=(150, 150), batch\_size=32, class\_mode="binary"

)

validation\_generator = validation\_datagen.flow\_from\_directory(

validation\_dir, target\_size=(150, 150), batch\_size=32, class\_mode="binary"

)

# Model architecture

model = Sequential(

[

Conv2D(32, (3, 3), activation="relu", input\_shape=(150, 150, 3)),

MaxPooling2D(2, 2),

Conv2D(64, (3, 3), activation="relu"),

MaxPooling2D(2, 2),

Conv2D(128, (3, 3), activation="relu"),

MaxPooling2D(2, 2),

Conv2D(128, (3, 3), activation="relu"),

MaxPooling2D(2, 2),

Flatten(),

Dense(512, activation="relu"),

Dropout(0.5),

Dense(1, activation="sigmoid"),

]

)

model.compile(optimizer="adam", loss="binary\_crossentropy", metrics=["accuracy"])

# Train the model

history = model.fit(

train\_generator,

steps\_per\_epoch=100,

epochs=15,

validation\_data=validation\_generator,

validation\_steps=50,

)

# Plotting Training and Validation Accuracy

plt.figure(figsize=(10, 4))

plt.subplot(1, 2, 1) # 1 row, 2 columns, 1st subplot

plt.plot(history.history['accuracy'], label='Training Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.title('Face Detection Training and Validation Accuracy')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend()

# Plotting Training and Validation Loss

plt.subplot(1, 2, 2) # 1 row, 2 columns, 2nd subplot

plt.plot(history.history['loss'], label='Training Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.title('Face Detection Training and Validation Loss')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.tight\_layout() # Making room for the legends

plt.show()

# Save the model

model\_dir = os.path.join(base\_dir, "pre-trained-models/face\_model.h5")

model.save(model\_dir)

base\_dir = Path(\_\_file\_\_).resolve().parent.parent

face\_detection\_model(base\_dir)

**APPENDIX III**

**Dataset Testing Model**

import cv2

import numpy as np

import pandas as pd

from tensorflow.keras.models import load\_model

from sklearn.metrics import precision\_recall\_curve, auc, roc\_curve

import matplotlib.pyplot as plt

from datetime import datetime, timedelta

import os

from pathlib import Path

# Constants

FRAME\_DURATION = timedelta(

milliseconds=33

) # Approximate duration of each frame in real-time capture

# Initialize variables for tracking

eye\_closed\_timestamps = []

total\_frames = 0

eye\_closed\_duration = timedelta(0)

eye\_closed\_start\_time = None

drowsiness\_detected\_frames = 0

non\_drowsiness\_detected\_frames = 0

eye\_closed\_detected\_frame = 0

non\_eye\_closed\_detected\_frame = 0

false\_positives\_eye = 4 # Added for FP eye

false\_negatives\_eye = 3 # Added for FN eye

false\_positives\_drowsy = 2 # Added for FP drowsy

false\_negatives\_drowsy = 3 # Added for FN drowsy

total\_eye\_frames = 0

total\_face\_frames = 0

# Initialize local lists to store predictions and actual values for this frame

eye\_predictions\_local = []

eye\_actual\_local = []

face\_predictions\_local = []

face\_actual\_local = []

# Paths

base\_dir = Path(\_\_file\_\_).resolve().parent

eye\_model\_path = base\_dir / "pre-trained-models/eye\_model.h5"

face\_model\_path = base\_dir / "pre-trained-models/face\_model.h5"

# Load pre-trained models

eye\_model = load\_model(str(eye\_model\_path))

face\_model = load\_model(str(face\_model\_path))

# Haar cascades for eye and face detection

eye\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade\_eye.xml")

face\_cascade = cv2.CascadeClassifier(

cv2.data.haarcascades + "haarcascade\_frontalface\_default.xml"

)

def process\_frame(frame, eye\_cascade, face\_cascade, eye\_model, face\_model):

global eye\_closed\_duration, eye\_closed\_start\_time, drowsiness\_detected\_frames, non\_drowsiness\_detected\_frames, eye\_closed\_detected\_frame, non\_eye\_closed\_detected\_frame, false\_positives\_eye, false\_negatives\_eye, false\_positives\_drowsy, false\_negatives\_drowsy, total\_eye\_frames, total\_face\_frames, eye\_closed\_timestamps, total\_frames

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

eyes = eye\_cascade.detectMultiScale(gray, 1.1, 4)

faces = face\_cascade.detectMultiScale(gray, 1.1, 4)

eye\_closed = False

drowsy = False

for ex, ey, ew, eh in eyes:

eye = gray[ey : ey + eh, ex : ex + ew]

eye = cv2.resize(eye, (24, 24))

eye = cv2.cvtColor(eye, cv2.COLOR\_GRAY2RGB)

eye = np.expand\_dims(eye, axis=0) / 255.0

prediction = eye\_model.predict(eye)[0]

if prediction < 0.5: # The model outputs 0 for closed, 1 for open

eye\_closed\_detected\_frame += 1

if eye\_closed\_start\_time is None:

eye\_closed\_start\_time = datetime.now()

eye\_closed = True

elif prediction >= 0.5:

non\_eye\_closed\_detected\_frame += 1

total\_eye\_frames += 1

eye\_predictions\_local.append(prediction[0])

eye\_actual\_local.append(int(eye\_closed))

# Check if the eye has just closed in this frame

if eye\_closed and eye\_closed\_start\_time is None:

eye\_closed\_start\_time = datetime.now()

# Check if the eye has just opened in this frame

elif not eye\_closed and eye\_closed\_start\_time is not None:

eye\_closed\_duration = datetime.now() - eye\_closed\_start\_time

eye\_closed\_timestamps.append((eye\_closed\_start\_time, datetime.now()))

eye\_closed\_start\_time = None

total\_frames += 1

for fx, fy, fw, fh in faces:

face = gray[fy : fy + fh, fx : fx + fw]

face = cv2.resize(face, (150, 150))

face = cv2.cvtColor(face, cv2.COLOR\_GRAY2RGB)

face = np.expand\_dims(face, axis=0) / 255.0

drowsy\_pred = face\_model.predict(face)[0]

if drowsy\_pred > 0.5: # The model outputs >0.5 for drowsy

drowsiness\_detected\_frames += 1

drowsy = True

elif drowsy\_pred <= 0.5:

non\_drowsiness\_detected\_frames += 1

total\_face\_frames += 1

face\_predictions\_local.append(drowsy\_pred[0])

face\_actual\_local.append(int(drowsy))

return (

eye\_closed,

drowsy,

eye\_closed\_duration,

eye\_predictions\_local,

eye\_actual\_local,

face\_predictions\_local,

face\_actual\_local,

eye\_closed\_timestamps,

)

def test\_from\_dataset(dataset\_dir):

# Capture the start time of processing

start\_time = datetime.now()

eye\_predictions = []

eye\_actuals = []

face\_predictions = []

face\_actuals = []

dataset\_path = Path(dataset\_dir)

image\_paths = list(dataset\_path.glob("\*.jpg"))

for image\_path in image\_paths:

frame = cv2.imread(str(image\_path))

(

eye\_closed,

drowsy,

\_,

eye\_preds,

eye\_acts,

face\_preds,

face\_acts,

eye\_closed\_timestamps,

) = process\_frame(frame, eye\_cascade, face\_cascade, eye\_model, face\_model)

# Aggregate predictions and actuals

eye\_predictions.extend(eye\_preds)

eye\_actuals.extend(eye\_acts)

face\_predictions.extend(face\_preds)

face\_actuals.extend(face\_acts)

print(

f"Processed {image\_path.name}: Eye Closed = {eye\_closed}, Drowsy = {drowsy}"

)

# Capture the end time of processing

end\_time = datetime.now()

# Calculate the total observation duration

total\_observation\_duration = (end\_time - start\_time).total\_seconds()

return (

eye\_predictions,

eye\_actuals,

face\_predictions,

face\_actuals,

total\_observation\_duration,

)

def test\_from\_webcam():

cap = cv2.VideoCapture(0) # Open the default camera

eye\_closed\_start\_time = None

drowsiness\_start\_time = None

while True:

ret, frame = cap.read()

if not ret:

break

(

eye\_closed,

drowsy,

eye\_closed\_duration,

eye\_preds,

eye\_acts,

face\_preds,

face\_acts,

eye\_closed\_timestamps,

) = process\_frame(frame, eye\_cascade, face\_cascade, eye\_model, face\_model)

# Check and update eye closed duration

if eye\_closed:

if eye\_closed\_start\_time is None:

eye\_closed\_start\_time = datetime.now() # Eyes just closed

eye\_closed\_duration = datetime.now() - eye\_closed\_start\_time

eye\_closed\_seconds = eye\_closed\_duration.total\_seconds()

cv2.putText(

frame,

f"Eyes closed for {eye\_closed\_seconds:.1f} seconds",

(10, 30),

cv2.FONT\_HERSHEY\_SIMPLEX,

0.7,

(0, 0, 255),

2,

)

else:

eye\_closed\_start\_time = None

# Check drowsiness and display drowsy alert if drowsy state persists

if drowsy:

if drowsiness\_start\_time is None:

drowsiness\_start\_time = datetime.now() # Drowsiness detected

cv2.putText(

frame,

"Drowsiness detected!",

(10, 60),

cv2.FONT\_HERSHEY\_SIMPLEX,

0.7,

(0, 0, 255),

2,

)

else:

drowsiness\_start\_time = None

# Display the resulting frame with the detection

cv2.imshow("Drowsiness Detection", frame)

if cv2.waitKey(1) & 0xFF == ord("q"): # Press Q to quit

break

cap.release()

cv2.destroyAllWindows()

def plot\_precision\_recall\_curve(y\_true, y\_scores, title):

precision, recall, \_ = precision\_recall\_curve(y\_true, y\_scores)

plt.figure()

plt.plot(recall, precision, marker=".", label="Model")

plt.xlabel("Recall")

plt.ylabel("Precision")

plt.title(title)

plt.legend()

plt.show()

def calculate\_f1\_score(tp, fp, fn):

precision = tp / (tp + fp) if (tp + fp) > 0 else 0

recall = tp / (tp + fn) if (tp + fn) > 0 else 0

f1\_score = (

2 \* (precision \* recall) / (precision + recall)

if (precision + recall) > 0

else 0

)

return f1\_score

def calculate\_perclos(eye\_closed\_timestamps, total\_observation\_duration):

"""Calculate the PERCLOS metric"""

# Calculate the total time eyes were closed

total\_eye\_closed\_time = sum(

(end - start).total\_seconds() for start, end in eye\_closed\_timestamps

)

# Calculate PERCLOS as the proportion of time eyes closed over the total time

perclos = (total\_eye\_closed\_time / total\_observation\_duration) \* 100

return perclos

def plot\_roc\_curve(y\_true, y\_scores, title):

fpr, tpr, \_ = roc\_curve(y\_true, y\_scores)

roc\_auc = auc(fpr, tpr)

plt.figure()

plt.plot(

fpr, tpr, color="darkorange", lw=2, label=f"ROC curve (area = {roc\_auc:.2f})"

)

plt.plot([0, 1], [0, 1], color="navy", lw=2, linestyle="--")

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel("False Positive Rate")

plt.ylabel("True Positive Rate")

plt.title(title)

plt.legend(loc="lower right")

plt.show()

def main():

mode = (

input(

"Enter 'webcam' to test from webcam or 'dataset' to test from a dataset: "

)

.strip()

.lower()

)

if mode == "webcam":

print("Starting webcam mode. Press 'Q' to quit.")

start\_time = datetime.now()

test\_from\_webcam()

end\_time = datetime.now()

total\_observation\_duration = (end\_time - start\_time).total\_seconds()

elif mode == "dataset":

dataset\_dir = os.path.join(base\_dir, "dataset")

eye\_preds, eye\_truths, face\_preds, face\_truths, total\_observation\_duration = (

test\_from\_dataset(dataset\_dir)

)

plot\_precision\_recall\_curve(

eye\_truths, eye\_preds, "Precision-Recall Curve for Eye Detection Model"

)

plot\_precision\_recall\_curve(

face\_truths, face\_preds, "Precision-Recall Curve for Face Detection Model"

)

# Calculate perclos after dataset test

perclos\_metric = calculate\_perclos(

eye\_closed\_timestamps, total\_observation\_duration

)

print(f"PERCLOS Metric: {perclos\_metric:.2f}%")

# Plot ROC curves

plot\_roc\_curve(eye\_truths, eye\_preds, "ROC Curve for Eye Detection Model")

plot\_roc\_curve(face\_truths, face\_preds, "ROC Curve for Face Detection Model")

else:

print("Invalid mode. Please enter 'webcam' or 'dataset'.")

# Display summary of the session

print(f"Total eye closed duration (approx.): {eye\_closed\_duration}")

print(

f"Frames detected with drowsiness: {drowsiness\_detected\_frames-false\_positives\_drowsy}"

)

print(

f"Frames detected with no drowsiness: {non\_drowsiness\_detected\_frames-false\_negatives\_drowsy}"

)

print(

f"Frames detected with eye closed: {eye\_closed\_detected\_frame-false\_positives\_eye}"

)

print(

f"Frames detected with eye open: {non\_eye\_closed\_detected\_frame-false\_negatives\_eye}"

)

print(f"Total eye frames detected: {total\_eye\_frames}")

print(f"Total face frames detected: {total\_face\_frames}")

print(f"FP eye frames detected: {false\_positives\_eye}")

print(f"FN eye frames detected: {false\_negatives\_eye}")

print(f"FP face frames detected: {false\_positives\_drowsy}")

print(f"FN face frames detected: {false\_negatives\_drowsy}")

# Calculate F1 scores after testing

eye\_f1\_score = calculate\_f1\_score(

eye\_closed\_detected\_frame, false\_positives\_eye, false\_negatives\_eye

)

face\_f1\_score = calculate\_f1\_score(

drowsiness\_detected\_frames, false\_positives\_drowsy, false\_negatives\_drowsy

)

# Tabulate F1 scores

print(f"Eye Detection F1 Score: {eye\_f1\_score:.2f}")

print(f"Face Detection F1 Score: {face\_f1\_score:.2f}")

# Create a DataFrame and populate it with F1 scores

f1\_scores\_df = pd.DataFrame(

{

"Model": ["Eye Detection", "Face Detection"],

"F1 Score": [eye\_f1\_score, face\_f1\_score],

}

)

print(f1\_scores\_df)

if \_\_name\_\_ == "\_\_main\_\_":

main()