Drowsiness Detection

A Project Work Report

Submitted in the partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE WITH SPECIALIZATION IN

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Submitted by:

RUCHI TIWARI 22BAI70283

VISHU 22BAI70264

NISHU 22BAI70365

Under the Supervision of:

Mr. Sant Kumar Maurya

CHANDIGARH UNIVERSITY, GHARUAN, MOHALI - 140413, PUNJAB

Table of Contents

Abstract

Drowsy driving remains a significant concern for road safety, leading to numerous accidents and fatalities worldwide. To address this issue, a sophisticated drowsiness detection device has been developed specifically for car drivers. This device integrates biometric sensors, advanced machine learning algorithms, real-time monitoring capabilities, and customizable alert mechanisms to accurately assess the driver's level of drowsiness and mitigate the risk of accidents. The device incorporates various biometric sensors, including electroencephalography (EEG), electromyography (EMG), and electrooculography (EOG), to continuously monitor the driver's physiological signals such as brainwave patterns, muscle activity, and eye movements. These signals are analysed in real-time using machine learning algorithms trained on extensive datasets of drowsy and alert states, enabling precise detection of drowsiness. The device offers a user-friendly interface displayed on the car dashboard or mobile app, providing drivers with insights into their drowsiness levels and recommendations for rest breaks. It also logs drowsiness-related data for post-journey analysis, facilitating pattern identification and algorithm refinement. The drowsiness detection device for car drivers represents a comprehensive solution to combat drowsy driving. By leveraging cutting-edge technology and customizable features.

Keywords- Drowsiness, Supervised Learning, Unsupervised Learning, Machine Learning.

1. INTRODUCTION

In an era marked by relentless advancements in technology and a parallel rise in the demands of modern life, the safety of road transportation stands as a paramount concern. Amidst the myriad challenges that confront road users, one insidious threat looms large: drowsy driving. The peril of nodding off at the wheel, even for a fleeting moment, can have catastrophic consequences, leading to accidents, injuries, and loss of precious lives. Recognizing the urgency of addressing this pressing issue, our project endeavors to delve into the realm of drowsiness detection, seeking innovative solutions to safeguard road users and enhance overall road safety.

At the heart of our project lies a fundamental commitment to leveraging technology as a catalyst for change. We understand that combating drowsy driving necessitates a multifaceted approach that integrates cutting-edge advancements with a deep understanding of human physiology and behavior. With this vision in mind, our project embarks on a journey to explore the intricacies of drowsiness detection, drawing upon a diverse array of methodologies and technologies to develop a comprehensive solution.

Central to our endeavor is the exploration of physiological signals and behavioral cues that betray the onset of drowsiness. We recognize that the human body exhibits subtle yet telltale signs when fatigue begins to set in: from changes in brain activity and heart rate to alterations in eye movement and posture. Through meticulous research and experimentation, our project aims to unravel the intricacies of these physiological and behavioral indicators, seeking to decode the language of drowsiness and develop robust algorithms capable of discerning its earliest manifestations.

In our pursuit of an effective drowsiness detection system, we confront a myriad of challenges and considerations. We grapple with questions of sensor placement and sensitivity, weighing the trade-offs between accuracy and user comfort. We delve into the intricacies of data processing and analysis, striving to distill meaningful insights from the deluge of physiological data captured in real-time. Moreover, we confront the ethical implications of drowsiness detection, grappling with issues of privacy, consent, and the responsible use of technology in safeguarding human lives.

As we navigate this complex landscape, our project remains steadfast in its commitment to innovation and excellence. We draw inspiration from the rich tapestry of interdisciplinary research, tapping into insights from fields as diverse as neuroscience, engineering, and psychology. Collaborating with experts and stakeholders across various domains, we harness the collective wisdom of the scientific community to inform our approach and refine our methodologies.

Yet, our project is more than a mere exercise in technical prowess; it is a testament to the power of human ingenuity and collaboration. We recognize that the challenge of drowsiness detection transcends the confines of any single discipline or institution. It requires a concerted effort, a shared commitment to advancing the frontiers of knowledge, and a collective resolve to make our roads safer for all.

As we embark on this journey, we invite fellow researchers, practitioners, and stakeholders to join us in our quest for safer roads and brighter futures. Together, let us harness the power of technology to confront the scourge of drowsy driving and pave the way for a world where every journey is a safe and enjoyable one.

As we conclude this synopsis, we recognize that our work represents not just a scientific endeavor, but a moral imperative. The lives saved, injuries prevented, and families spared from the devastation of road accidents are the ultimate measures of our success. We extend our gratitude to the countless individuals and organizations who have supported us along the way, from mentors and collaborators to funding agencies and community advocates. Your contributions have been invaluable in shaping the trajectory of our project and inspiring us to push the boundaries of innovation.

Looking ahead, we are committed to continuing our efforts in the pursuit of safer roads and more resilient communities. We recognize that the journey towards eliminating drowsy driving is an ongoing one, requiring sustained dedication, collaboration, and innovation. As we forge ahead, we remain guided by the belief that through collective action and unwavering resolve, we can make meaningful strides towards a safer, more sustainable future for all road users.

In closing, we invite you to join us in our mission to create a world where every journey is characterized by safety, security, and peace of mind. Together, let us harness the power of technology, innovation, and compassion to build a future where drowsy driving is nothing more than a distant memory. Thank you for your support, and we look forward to embarking on this journey together.

1.1 Problem Definition

The primary objective is to develop a drowsiness detection device capable of accurately assessing the level of driver fatigue in real-time and alerting the driver to mitigate the risk of accidents caused by drowsy driving. The device must address the following key challenges:

Accurate Detection: The device must accurately detect the onset and severity of driver drowsiness, considering various physiological and behavioral indicators, such as EEG, EMG, and EOG signals.

Real-time Monitoring: Real-time monitoring of driver drowsiness is essential to provide timely alerts and interventions. The device should continuously analyze data from biometric sensors and promptly identify signs of fatigue.

Customizable Alert Mechanism: The device should feature customizable alert mechanisms tailored to the severity of drowsiness detected. Alerts must be effective in capturing the driver's attention without causing distraction or discomfort.

Integration with Vehicle Systems: Seamless integration with existing vehicle systems is crucial to enable automatic activation of safety features in response to detected drowsiness. This integration enhances overall driving safety and augments the driver's ability to navigate challenging situations.

User Interface and Feedback: An intuitive user interface accessible via the car dashboard or a mobile application should provide drivers with insights into their drowsiness levels and driving behavior. The interface should offer personalized recommendations for fatigue management and encourage proactive engagement.

Privacy and Security: Protecting the privacy and security of driver data is paramount. The device must adhere to stringent privacy regulations and employ robust encryption protocols to safeguard sensitive biometric information.

Scalability and Accessibility: The device should be scalable and accessible to a wide range of drivers, accommodating diverse driving environments, vehicle types, and user preferences.

Objectives:

- Develop robust biometric sensors capable of accurately capturing physiological signals associated with drowsiness.
- Implement advanced machine learning algorithms to analyze biometric data and accurately detect drowsiness patterns.
- Design a customizable alert mechanism that effectively notifies drivers of their drowsiness levels without causing distraction.
- Integrate the device seamlessly with existing vehicle systems to enhance driving safety and responsiveness.
- Create an intuitive user interface for drivers to access real-time insights into their drowsiness levels and receive personalized recommendations.
- Ensure compliance with privacy regulations and implement stringent security measures to protect driver data.
- Conduct rigorous testing and validation to assess the effectiveness and reliability of the drowsiness detection device across diverse driving scenarios and user demographics.

1.2 Problem Overview

Drowsy driving presents a significant and persistent threat to road safety, contributing to a substantial number of accidents, injuries, and fatalities worldwide. Despite concerted efforts to raise awareness and enact regulations, the problem persists, necessitating innovative solutions to mitigate its impact. The problem overview provides a comprehensive understanding of the challenges posed by drowsy driving and highlights the urgent need for effective intervention strategies.

The overarching problem of drowsy driving presents a multifaceted challenge that demands a nuanced understanding of its complexities. It is not merely a matter of individuals experiencing fatigue while behind the wheel but rather a systemic issue intertwined with factors such as societal norms, lifestyle choices, work schedules, and the design of transportation systems. This comprehensive view illuminates the urgent need for interventions that address the root causes of drowsy driving while also considering the diverse contexts in which it occurs.

Moreover, the persistence of drowsy driving as a widespread concern underscores the inadequacy of traditional approaches in tackling this issue. While educational campaigns and legislative measures have been instrumental in raising awareness and establishing guidelines, they have not been fully effective in curbing the incidence of drowsy driving-related incidents. As such, there is a compelling imperative to explore novel avenues and deploy innovative technologies and methodologies to complement existing efforts and enhance their efficacy.

In light of these challenges, a thorough problem overview serves as a crucial foundation for devising and implementing effective intervention strategies. By gaining a comprehensive understanding of the underlying causes, risk factors, and consequences of drowsy driving, stakeholders are better equipped to develop targeted interventions that address the specific needs and circumstances of affected individuals and communities. This holistic approach not only facilitates the identification of high-impact intervention points but also ensures that resources are allocated judiciously to maximize their effectiveness.

1.3 Hardware Specification

Hardware Specifications for the Drowsiness Detection System:

Integration of Vehicle Sensors:

Steering Wheel Sensors: These sensors, mounted on the steering wheel, are designed to detect alterations in steering behavior, such as erratic movements or prolonged periods of inactivity, which often signify driver drowsiness.

Accelerometer and Gyroscope: Embedded within the vehicle, these inertial sensors capture motion data and identify deviations from standard driving patterns, such as abrupt deceleration or swerving.

Camera System: Utilizing in-cabin cameras, this component captures facial expressions and movements, facilitating visual monitoring of driver alertness and the identification of drowsiness-related indicators like drooping eyelids or head nodding.

Alerting Mechanisms:

Audio Output: Incorporating built-in speakers or audio output features, the system delivers auditory alerts and warnings to the driver. This aspect supports customizable audio cues with varying intensity levels to cater to individual preferences.

Visual Display: Employing a visual display unit, such as an LED display or heads-up display (HUD), the system presents visual alerts and notifications within the driver's line of sight. The display is designed to be bright, clear, and easily visible under diverse lighting conditions.

Power Supply:

Vehicle Power Adapter: This component consists of a power adapter compatible with the vehicle's electrical system, providing a continuous power supply to the drowsiness detection system. It is engineered to accommodate a broad input voltage range and incorporates surge protection mechanisms to mitigate risks associated with power fluctuations.

These hardware specifications have been meticulously designed to equip the drowsiness detection system with robust sensors, processing capabilities, alert mechanisms, connectivity options, and power supply, thereby enabling it to effectively monitor driver drowsiness and bolster road safety.

In essence, the hardware specifications for the drowsiness detection system encompass a comprehensive array of components tailored to the intricacies of monitoring driver alertness. The integration of vehicle sensors, including steering wheel sensors, accelerometers, and gyroscopes, enables the system to capture and analyze critical driving data indicative of drowsiness. Additionally, the incorporation of a camera system enhances the system's capabilities by providing visual cues for assessing the driver's state.

Furthermore, the alerting mechanisms, comprising audio output and visual displays, ensure timely and intuitive notifications to the driver, enabling prompt intervention in case of drowsiness. The audio output feature supports customizable alerts, catering to individual preferences, while the visual display unit offers clear and conspicuous notifications within the driver's field of vision.

Lastly, the reliable power supply facilitated by the vehicle power adapter ensures uninterrupted operation of the drowsiness detection system, thereby maintaining its efficacy throughout the duration of the drive. With these carefully crafted hardware specifications in place, the drowsiness detection system stands poised to enhance road safety by proactively addressing the risks associated with drowsy driving.

1.4 Software Specification

Python, in conjunction with the SciPy library, offers a robust platform for conducting signal processing tasks. The SciPy library encompasses a diverse array of signal processing functions and algorithms specifically designed for tasks such as filtering, spectral analysis, and feature extraction. This powerful combination enables developers to efficiently manipulate and analyze signals, thereby extracting meaningful insights and patterns.

When it comes to developing machine learning models aimed at classifying drowsiness states based on extracted features from physiological signals and vehicle data, TensorFlow and PyTorch emerge as prominent frameworks. These frameworks provide efficient implementations of deep learning algorithms and neural network architectures, making them well-suited for handling large datasets. TensorFlow and PyTorch streamline both the training and inference processes, facilitating the creation of accurate and robust models capable of discerning drowsiness states with high precision.

In addition to deep learning frameworks, Scikit-learn, a widely-used machine learning library in Python, plays a crucial role in the development of drowsiness detection systems. Scikit-learn offers a rich set of tools and algorithms for traditional machine learning tasks, including Support Vector Machines (SVM), Random Forests, and Gradient Boosting. Its versatility and efficiency make it particularly useful for tasks such as feature selection and model evaluation, complementing the capabilities of TensorFlow and PyTorch in the development pipeline.

In the domain of integrating with vehicle systems, specialized software tools designed for interfacing with the Controller Area Network (CAN) bus play a pivotal role. The CAN bus is a standard vehicle communication protocol that enables access to crucial vehicle data, including steering wheel angle, vehicle speed, and accelerometer readings. Popular software options such as CANalyzer or Vector CANoe facilitate seamless communication with onboard vehicle systems, ensuring the smooth exchange of data necessary for drowsiness detection.

Middleware software solutions, such as MQTT (Message Queuing Telemetry Transport) or WebSocket, further enhance the integration between the drowsiness detection system and vehicle systems. These protocols enable real-time communication and data exchange among different components of the system, ensuring timely responses to drowsiness-related cues detected from vehicle data. By leveraging middleware software, developers can ensure efficient and effective coordination between the drowsiness detection system and vehicle systems, thereby enhancing overall system performance and responsiveness.

Moreover, given the need to store unstructured or semi-structured data generated by the drowsiness detection system, NoSQL databases such as MongoDB or Cassandra emerge as viable options. These databases offer scalability and distributed storage capabilities, making them well-suited for accommodating large volumes of sensor readings, log files, and other data generated during the operation of the drowsiness detection system. By leveraging NoSQL databases, developers can efficiently manage and query large volumes of data, ensuring the robustness and reliability of the overall system.

In summary, Python, along with the SciPy library, serves as a versatile platform for signal processing tasks, enabling developers to efficiently manipulate and analyze signals. TensorFlow, PyTorch, and Scikit-learn provide powerful frameworks for developing machine learning models, while specialized software tools and middleware solutions facilitate integration with vehicle systems. Additionally, NoSQL databases offer scalable storage solutions for managing the vast amounts of data generated by the drowsiness detection system. Together, these technologies form a comprehensive ecosystem for building and deploying effective drowsiness detection systems tailored to enhance road safety.

2. RELATED WORK

Within this chapter, we embark on a comprehensive exploration of the evolution of drowsiness detection methodologies, with a primary focus on advancing both the precision and expediency of detection techniques. Initially, we provide an exhaustive review of conventional strategies historically employed for drowsiness detection, meticulously outlining their methodologies, strengths, and inherent limitations. This retrospective analysis offers valuable insights into the progression of drowsiness detection research and underscores the imperative for innovative approaches to address prevailing challenges.

Subsequently, we transition to an in-depth examination of contemporary methodologies that harness the power of deep learning paradigms. These cutting-edge techniques have emerged as frontrunners in the field, leveraging the inherent capabilities of neural networks to discern subtle cues indicative of drowsiness with unprecedented accuracy. Through a comprehensive evaluation of recent advancements, we elucidate the transformative potential of deep learning in revolutionizing the landscape of drowsiness detection.

Furthermore, we delve into the realm of deep learning model compression methods, a burgeoning area of research aimed at circumventing runtime constraints encountered during drowsiness detection. By succinctly summarizing the latest innovations in model compression, we elucidate how these strategies enable the seamless integration of sophisticated deep learning models into real-world applications, thereby enhancing the efficiency and scalability of drowsiness detection systems.

In essence, this chapter serves as a comprehensive compendium of past, present, and future methodologies in drowsiness detection, offering readers a nuanced understanding of the evolving landscape and the promising avenues for further exploration and innovation.

2.1. Conventional Approaches for Drowsiness Detection

Driving pattern analysis involves evaluating steering wheel movements or deviations from the lane's position, offering insights into a driver's alertness level. Krajweski et al. achieved an 86% accuracy in detecting drowsiness by correlating subtle steering adjustments with drowsiness indicators. Similarly, monitoring lane deviations provides valuable information about a driver's attentiveness, although these methods are influenced by factors such as driving expertise, road conditions, and vehicle characteristics.

In contrast, physiological sensors, such as EEG, ECG, and EOG, offer direct measurements of bodily responses to drowsiness. EEG signals, for instance, reflect brain activity, with changes in alpha, delta, and theta waves indicating varying levels of drowsiness. While EEG-based methods boast high accuracy, surpassing 90% in some studies, their intrusive nature, requiring multiple sensors attached to the body, can be disruptive for drivers. Despite efforts to develop non-intrusive alternatives, their accuracy often falls short compared to invasive methods.

Facial feature extraction via Computer Vision techniques presents a non-invasive alternative for drowsiness detection. By analyzing facial cues like eye closure, head movement, yawning frequency, and facial expressions, researchers can infer drowsiness levels. Danisman et al., for instance, utilized eyelid distance to categorize drowsiness levels, while others focused on mouth movements and yawning. The use of machine learning algorithms, such as the modified Viola-Jones object detection, enhances the accuracy of facial feature extraction methods.

Overall, drowsiness detection strategies encompass a diverse array of approaches, each with its strengths and limitations. While driving pattern analysis provides real-time insights into driver behavior, physiological sensors offer direct measures of drowsiness, albeit with intrusive drawbacks. On the other hand, facial feature extraction presents a non-invasive solution, but may require sophisticated algorithms for accurate interpretation. By understanding the nuances of each method, researchers can tailor drowsiness detection systems to suit specific applications, ultimately enhancing road safety.

2.2. Drowsiness Detection using Deep Learning

Recently, deep learning is widely used to resolve difficult problems which cannot be handled properly using conventional methods. Deep learning based on Convolutional Neural Networks (CNNs) makes a breakthrough especially for Computer Vision tasks such as image classification, object detection, emotion recognition, scene segmentation [10]-[13] etc.

Dwivedi et al. [14] adopted shallow CNNs for drowsy driver detection with accuracy of 78%. As the latest research, S. Park et al. [15] proposed a new architecture using three networks. In the first network, image feature is learnt by using AlexNet which consists of 5 CNNs and 3 FC layers [16].

16-layered VGG-FaceNet [17] is utilized to extract facial feature in the second network. The last network works to extract behavior features by using FlowImageNet [18]. As a result, 73% detection accuracy is achieved. Both [14] and [15] focused on improving drowsiness detection accuracy employing binary classification. In real-time applications, performance in terms of speed is also a crucial point.

In recent years, deep learning has emerged as a powerful tool for tackling complex problems that traditional methods struggle to address effectively. Convolutional Neural Networks (CNNs) have particularly revolutionized Computer Vision tasks, including image classification, object detection, emotion recognition, and scene segmentation.

Dwivedi et al. applied shallow CNNs to drowsy driver detection, achieving an accuracy of 78%. In contrast, S. Park et al. introduced a novel architecture consisting of three networks in their latest research. The first network utilized AlexNet, comprising 5 CNNs and 3 fully connected (FC) layers, to learn image features. The second network employed a 16-layered VGG-FaceNet to extract facial features, while the third network utilized FlowImageNet to extract behavioral features. This multi-network approach yielded a detection accuracy of 73%.

Both studies focused on enhancing drowsiness detection accuracy through binary classification. However, in real-time applications, speed is also a critical factor to consider alongside accuracy.

To summarize, recent advancements in deep learning, particularly using CNNs, have significantly improved drowsiness detection capabilities. While Dwivedi et al. achieved notable accuracy with shallow CNNs, S. Park et al. proposed a more intricate architecture utilizing multiple networks. Despite their successes, the challenge of balancing accuracy and speed remains paramount in real-world applications of drowsiness detection.

2.3. Compression Algorithms of Deep Learning Model

Although deep learning is powerful on various classification tasks, it is a burden to deploy deep learning algorithms to practical applications on embedded systems since model size of deep learning is generally large and high computational complexity is required. Therefore, in the recent years algorithms to reduce model size and improve speed have been proposed by using various ways [19]-[29].

Methods to reduce model size have been proposed in [19]-[23]. Generally, trained networks include redundant information, so some of weights can be discarded by applying pruning without accuracy drop. To reduce model size further, quantization techniques have been introduced such as bit-quantization. In bit-quantization, the least number of bits are utilized for representing information of model while minimizing accuracy loss. In some of researches, they adopted binary networks. Even though these works have advantages in terms of model size and speed, accuracy cannot be maintained because of the simplicity of binary operations [21][22].

Moreover, Low-rank decomposition has been proposed in [24] to decompose a tensor and reduce number of matrix operations.

R. Caruana et al. [26] introduced the concept of applying ensemble selection from libraries of models. According to this work, researchers in [25][27][28][29][30] developed learning algorithms by adopting knowledge distillation approach between two networks. A role of one is teacher and that of the other network is student.

Teacher networks are large and have high computation requirements, which can learn patterns from a large dataset. On the contrary student networks are small requiring less computation and can learn only from teacher network. Due to its smaller size, student network is suitable to be implemented on embedded devices and has capability to run at real time on portable devices. Hinton et al. [25] proposed how to transfer weights from teacher network to student network using knowledge distillation. As a result the student network can successfully learn from the teacher.

3. PROBLEM FORMULATION

Objective: The primary objective of this project is to design and implement a robust drowsiness detection system for car drivers to enhance road safety by mitigating the risks associated with drowsy driving.

Scope: The system will focus on detecting drowsiness in real-time using physiological signals, behavioural cues, and vehicle dynamics.

It will be applicable to various driving conditions, including day and night driving, short and long distances, and diverse weather conditions.

The system will aim to provide timely alerts to drivers and may include intervention mechanisms to prevent accidents.

Challenges:

Accurate Detection: Developing algorithms capable of accurately detecting drowsiness based on a combination of physiological signals (e.g., EEG, EMG, EOG), behavioural cues (e.g., eye closure, head nodding), and vehicle dynamics (e.g. steering behaviour)

Real-time Processing: Ensuring efficient processing and analysis of data streams in real-time to enable timely detection and alerting without introducing significant latency.

Customizable Alerts: Designing customizable alert mechanisms that effectively notify drivers of their drowsiness levels without causing distraction or discomfort, considering individual preferences and driving contexts.

Integration with Vehicle Systems: Integrating the detection system seamlessly with existing vehicle systems to enable automated responses and enhance driving safety without compromising vehicle performance or usability.

Approach:
Research existing drowsiness detection methods, algorithms, and technologies to identify best practices and potential areas for improvement.

Develop signal processing algorithms to analyse physiological signals and extract features
Indicative of drowsiness

Design machine learning models to classify drowsiness states based on extracted features and historical data.

Implement customizable alert mechanisms and intervention strategies to notify drivers and prevent accidents.

Integrate the detection system with vehicle systems and develop a user-friendly interface for driver interaction.

Outcome: The expected outcome of this project is a reliable and effective drowsiness detection system that enhances road safety by alerting drivers to their drowsiness levels and facilitating proactive intervention to prevent accidents caused by driver fatigue. The system will contribute to reducing the incidence of drowsy driving-related accidents and saving lives on the road.

4. OBJECTIVES

Objectives:

- Design and Development of the Drowsiness Detection System:
- Develop a comprehensive understanding of existing drowsiness detection techniques, algorithms, and technologies.
- Design the architecture and components of the drowsiness detection system, including sensor integration, data processing, alert mechanisms, and user interfaces.
- Define system requirements, specifications, and performance metrics to guide development.

Signal Processing and Feature Extraction:

- Implement signal processing algorithms to filter, preprocess, and extract features from physiological signals (EEG, EMG, EOG) and vehicle data (steering behaviour, vehicle dynamics).
- Identify relevant features indicative of drowsiness, such as spectral characteristics, muscle activity patterns, eye movements, and driving behaviour.

Machine Learning Model Development:

- Develop machine learning models to classify drowsiness states based on extracted features.
- Explore various machine learning techniques, including deep learning, ensemble methods, and traditional classifiers, to optimize detection accuracy and efficiency.
- Train and validate the models using labelled datasets collected from real-world driving scenarios.

Alert Mechanism Implementation:

- Design and implement customizable alert mechanisms to notify drivers of their drowsiness levels.
- Integrate auditory, visual, and tactile feedback cues to deliver timely alerts without causing distraction or discomfort to the driver.
- Incorporate adaptive alert strategies based on the severity of drowsiness detected and driver preferences.

5. METHODOLOGY

Drowsiness detection typically involves the use of various sensors and machine learning algorithms to analyze physiological and behavioral indicators associated with drowsiness. The methodology for drowsiness detection can be broken down into several key steps:

Data Acquisition:

- Sensors: Use sensors to collect data on physiological and behavioral parameters. Commonly used sensors include:
- Eye-Tracking Devices: Monitor eye movements and blink patterns.
- Electroencephalogram (EEG): Measure brainwave activity.

Electrocardiogram (ECG): Monitor heart rate variability.

- Accelerometers and Gyroscopes: Track head movements.
- Facial Expression Analysis: Capture facial features and expressions.
- Vehicle-based Sensors: Utilize vehicle data, such as steering wheel movements or lane departure information.

Data Preprocessing:

- Filtering and Noise Reduction: Clean the acquired data by filtering out noise and artifacts.
- Normalization: Normalize data to ensure consistency and comparability across different individuals and sessions.
- Feature Extraction: Extract relevant features from the raw data. Features may include blink frequency, eye closure duration, heart rate variability, etc.

Labelling:

- Annotate the dataset with labels indicating periods of alertness and drowsiness. This labelling is typically done using subjective measures, such as self-reported drowsiness or external observations.
- Model Training:
- Selection of ML Algorithm: Choose a suitable machine learning algorithm based on the nature of the data and the problem. Commonly used algorithms include Support Vector Machines (SVM), Random Forests, Neural Networks, or hybrid models.
- Feature Selection: Identify and select the most relevant features for training the model.
- Training Data: Split the dataset into training and validation sets. Train the model on the labelled training data to learn the patterns associated with drowsiness.

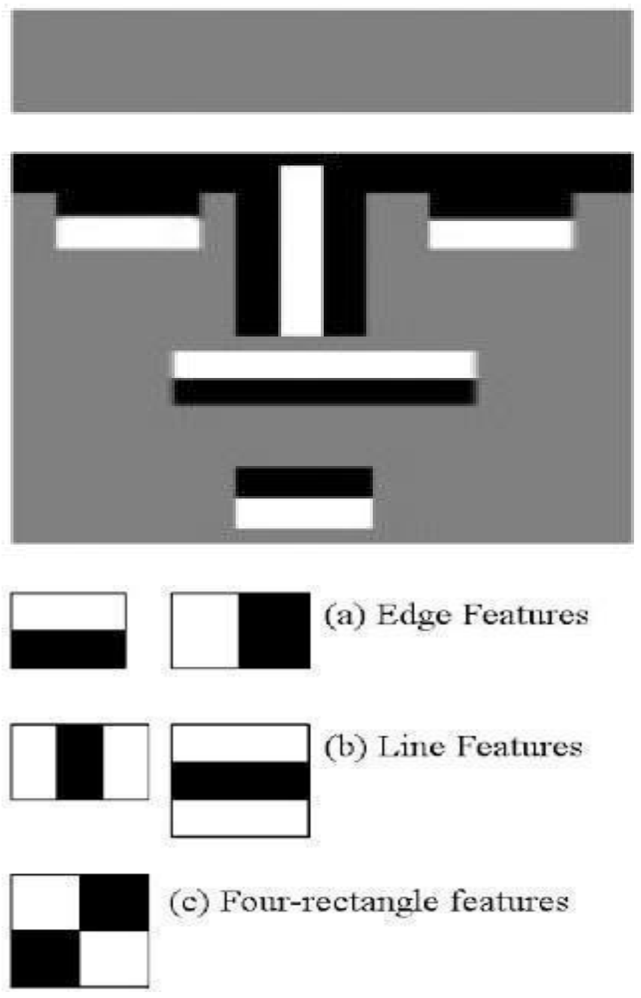
Model Evaluation:

- Validation: Assess the performance of the trained model on a separate validation dataset to ensure it generalizes well to new data.
- Metrics: Use metrics such as accuracy, precision, recall, and F1 score to evaluate the model's performance in detecting drowsiness.

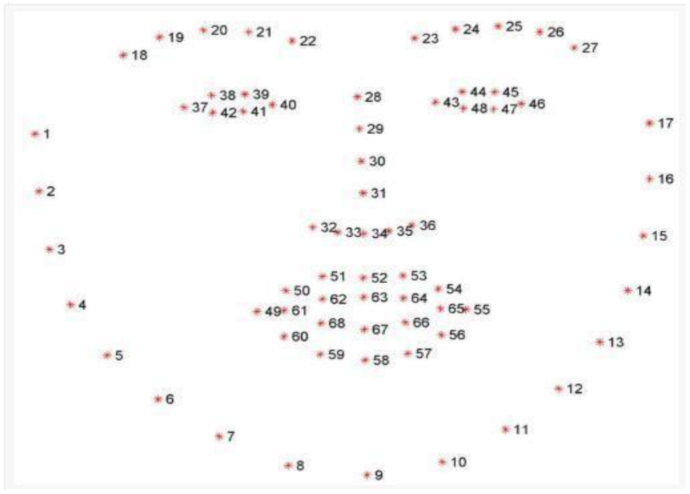
Real-time Detection:

- Implement the trained model in a real-time environment, such as a vehicle or a workplace setting.
- Continuously monitor incoming data and classify the current state as either alert or drowsy based on the learned patterns.
- Feedback and Intervention:
- Provide timely feedback or interventions when drowsiness is detected. This may involve visual or auditory alerts, haptic feedback, or other forms of intervention to prompt the individual to stay alert.

- Iterative Improvement:
- Continuously gather new data and update the model to improve its accuracy and adaptability over time.
- The specific methodology can vary based on the application, available sensors, and the context in which drowsiness detection is implemented. Additionally, the choice of features and machine learning algorithms may be influenced by the nature of the data and the desired level of real-time performance.



(A)


(B)

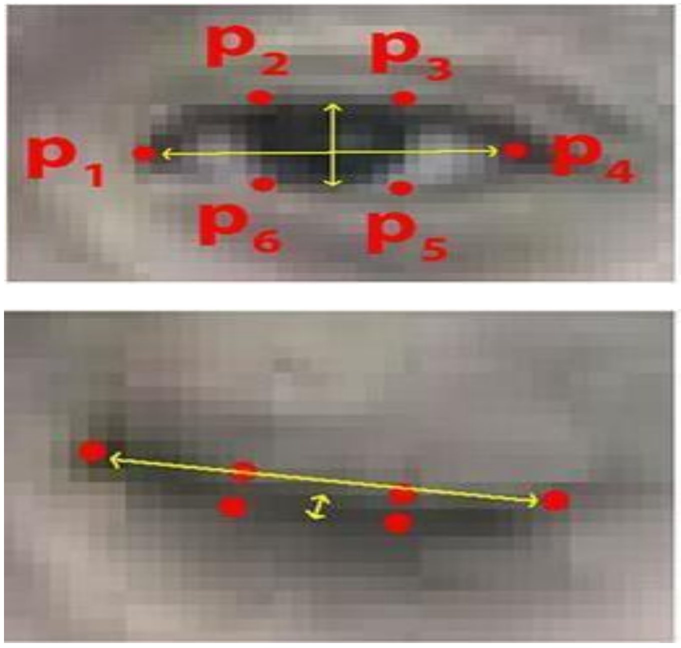


Fig.4.1: Eyes Feature Extraction

6. FLOWCHART AND REPRESENTATION

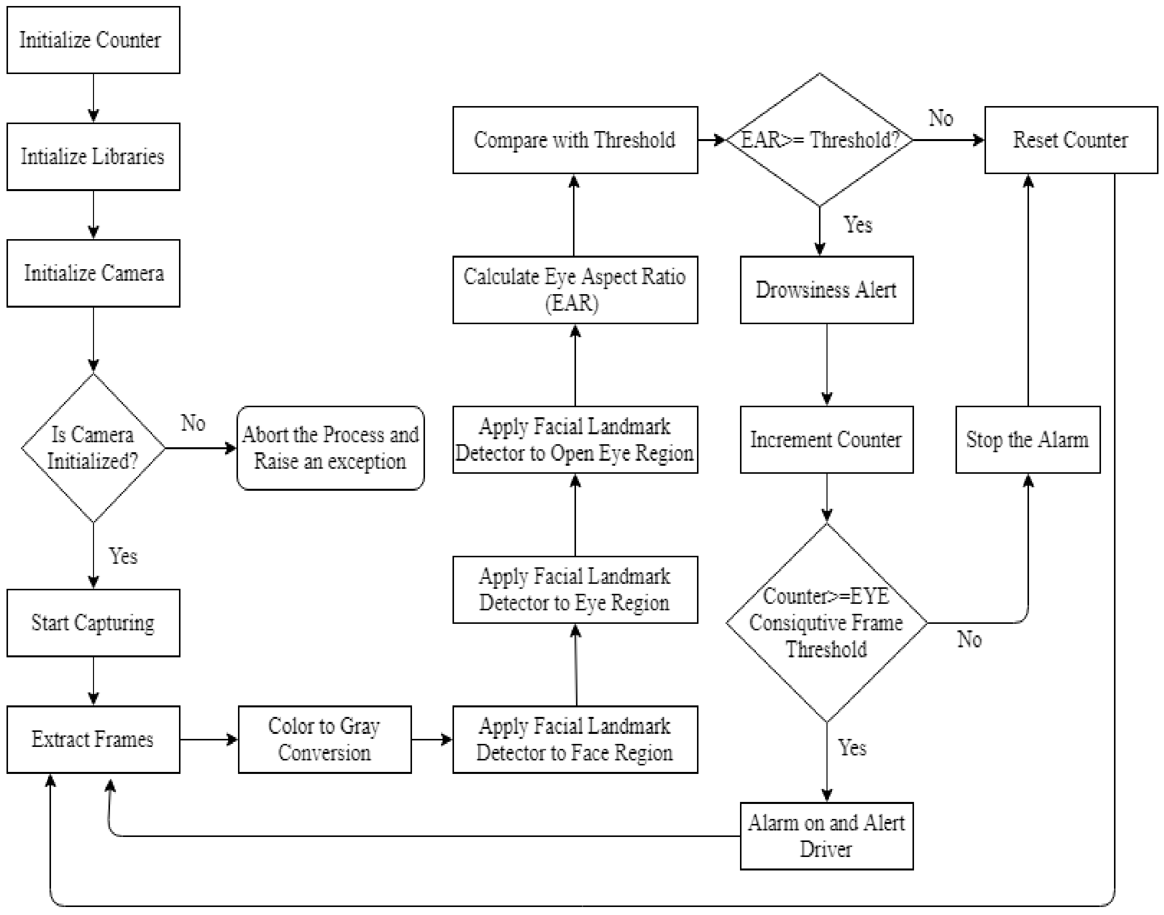

Initialize Counter
Intialize Libraries
Initialize Camera
Is Camera Initialized?
No
Abort the Process and Raise an exception
Yes
Start Capturing
Extract Frames
Color to Gray Conversion
Apply Facial Landmark Detector to Face Region
Apply Facial Landmark Detector to Eye Region
Apply Facial Landmark Detector to Open Eye Region
Calculate Eye Aspect Ratio (EAR)
Compare with Threshold
EAR>= Threshold?
No
Reset Counter
Yes
Drowsiness Alert
Increment Counter
Counter>=EYE Consiqutive Frame Threshold
No
Stop the Alarm
Yes
Alarm on and Alert Driver

Fig.5.1: Flowchart of system

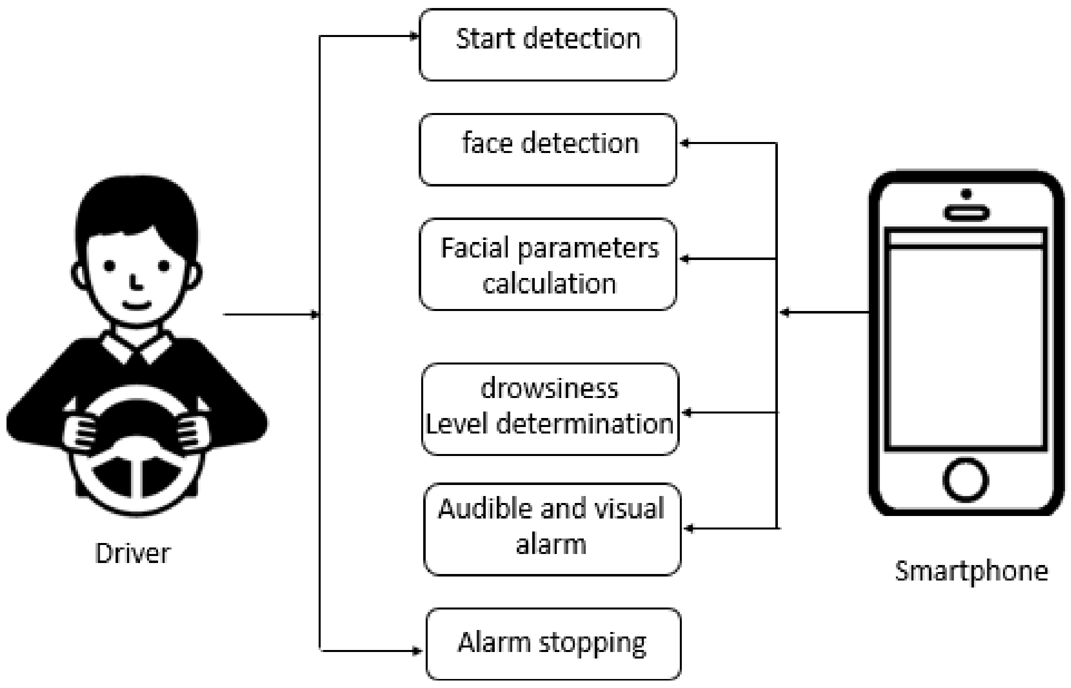


Fig.5.2: Working of system

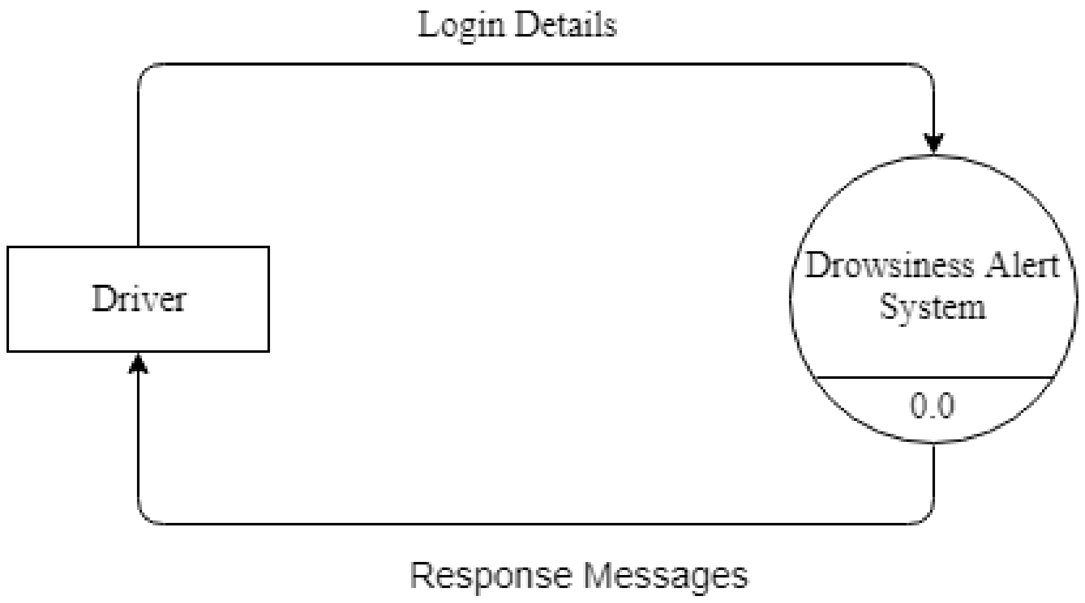


Fig.5.3: DFD Level 0

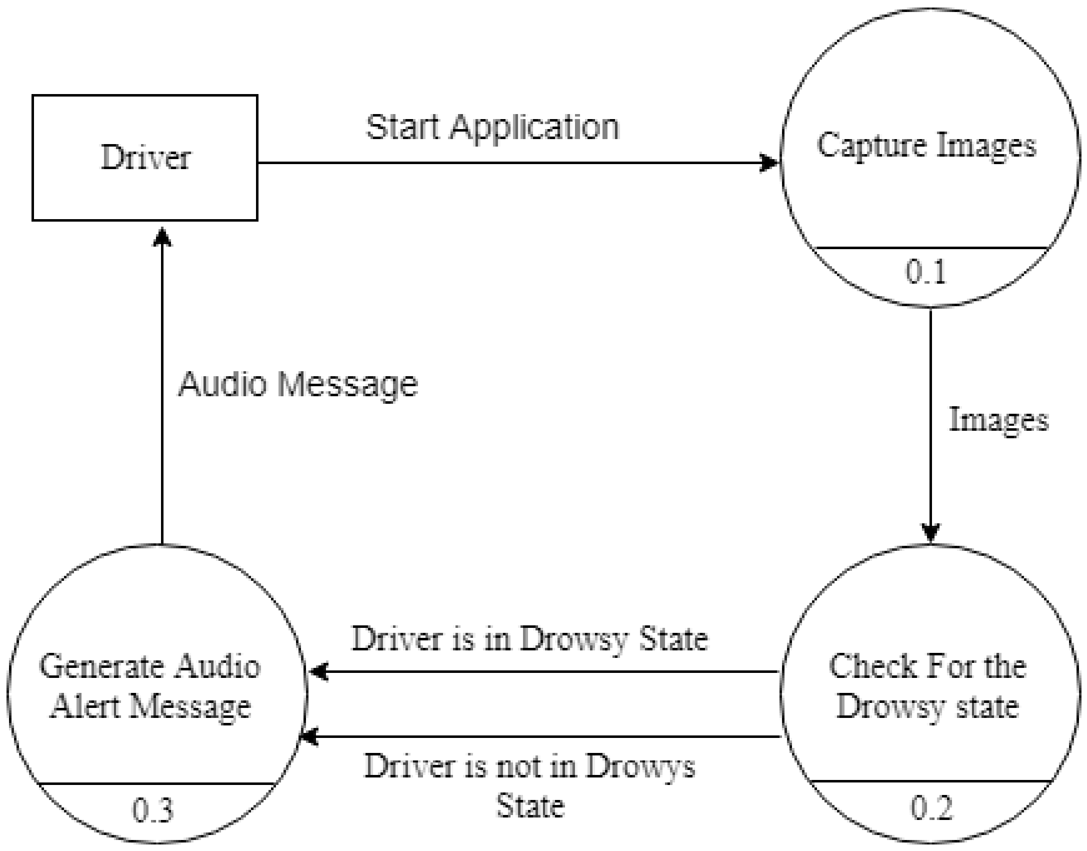

Driver
Start Application
Capture Images
0.1
Images
Audio Message
Check For the Drowsy state
0.2
Driver is in Drowsy State
Driver is not in Drowys State
Generate Audio Alert Message
0.3

Fig.5.4: DFD Level 1

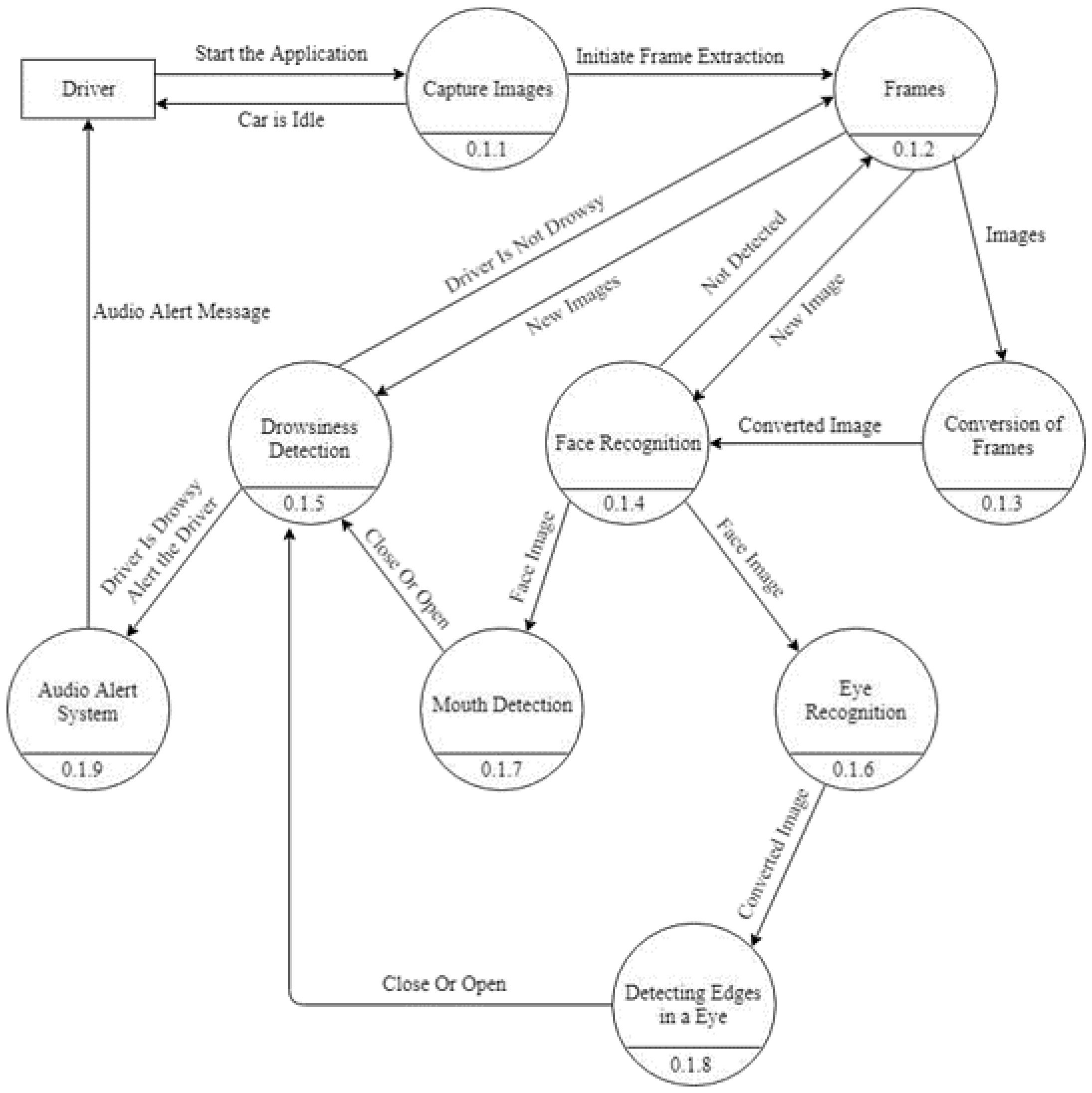


Fig.5.5: DFD Level 3

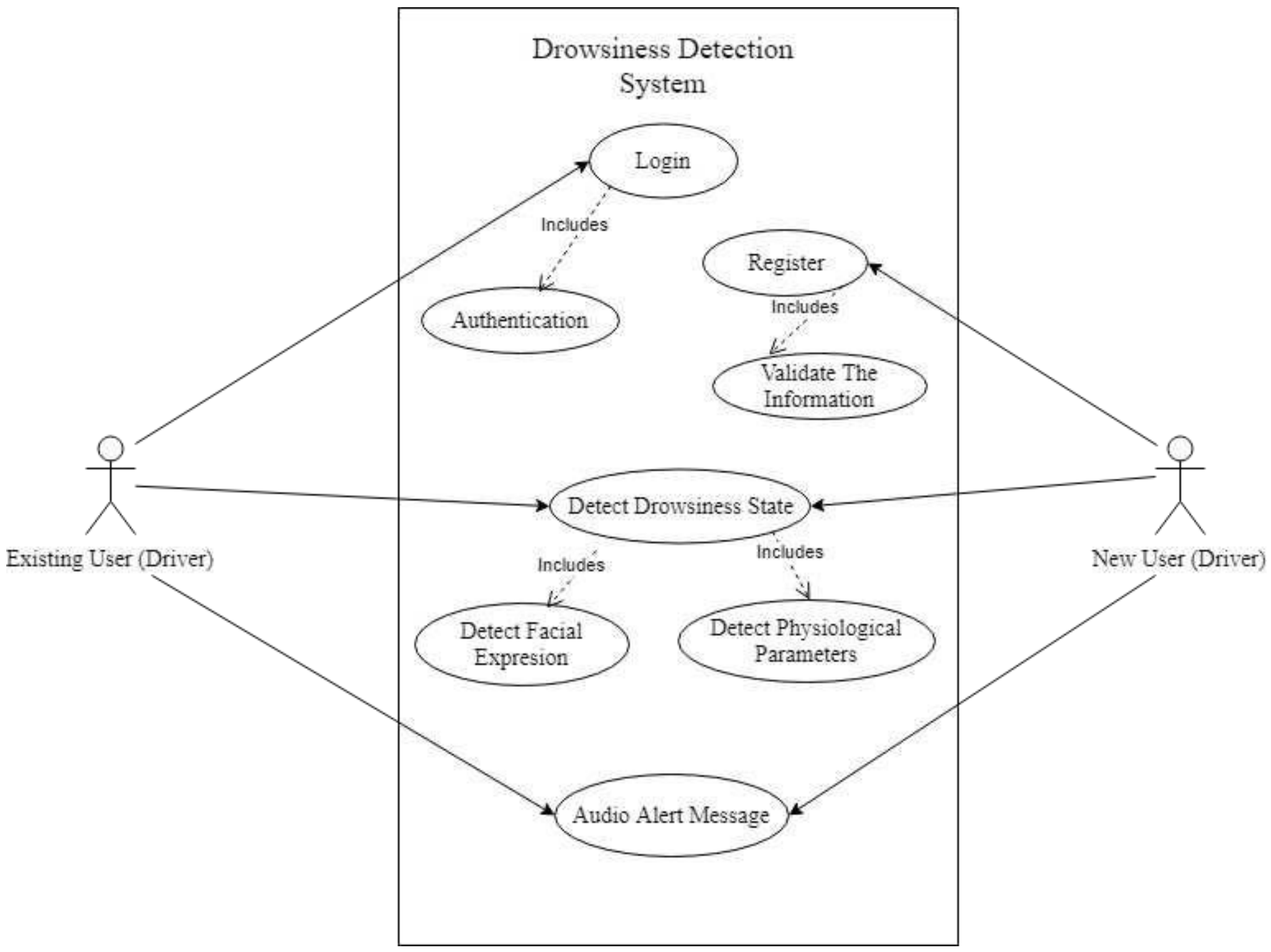


Fig 5.6 Use-Case Diagram

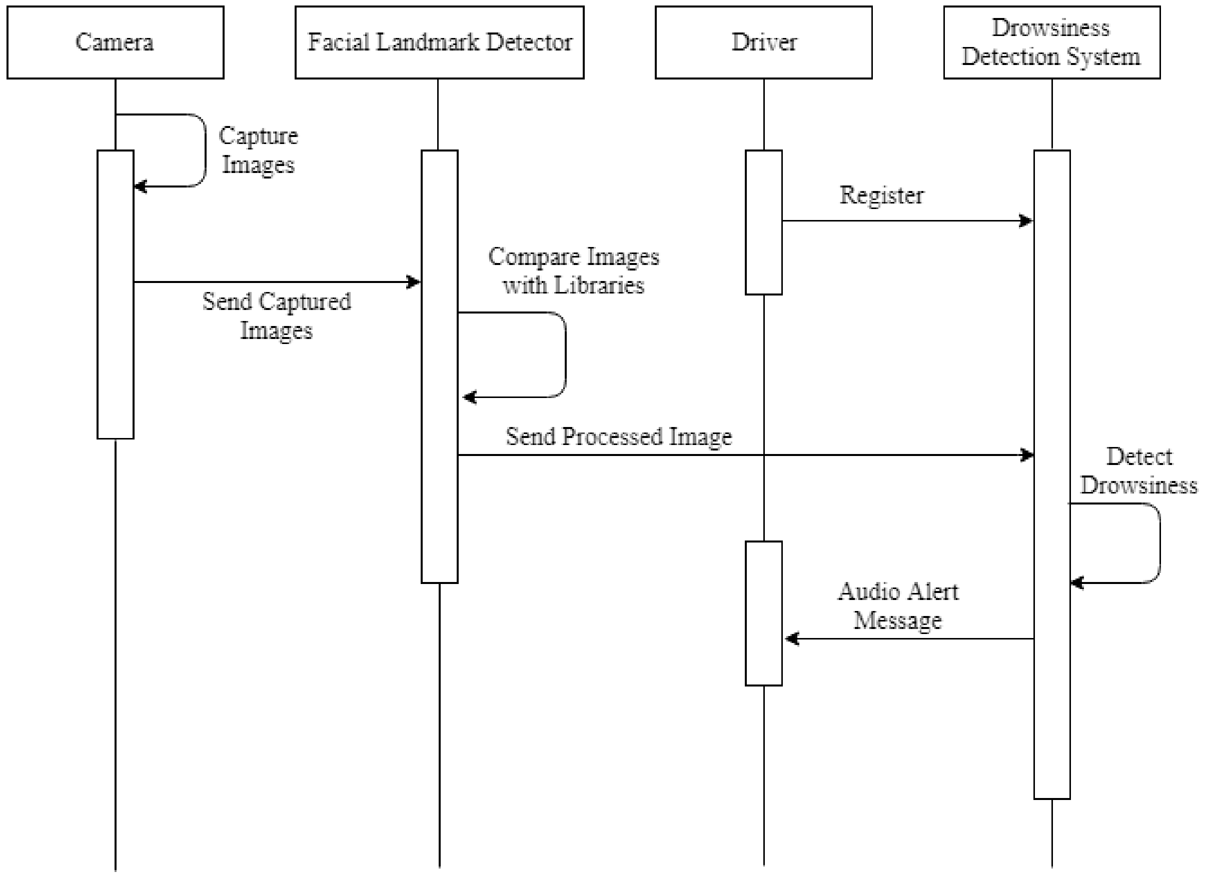


Fig.5.7: Sequence Diagram

7. EXPERIMENTAL SETUP

Hardware Setup:

- Biometric Sensors: Install EEG, EMG, and EOG sensors on the driver's head and face to capture physiological signals associated with drowsiness.
- Vehicle-Integrated Sensors: Mount sensors on the steering wheel, dashboard, and seat to measure vehicle dynamics, including steering behavior, acceleration, and seat movement.
- Camera System: Position cameras inside the vehicle to capture facial expressions, eye movements, and head positions of the driver.
- Alert Mechanisms: Integrate auditory, visual, and tactile feedback devices such as speakers, LED displays, and vibration modules into the vehicle for delivering alerts to the driver.

Software Setup:

- Signal Processing Software: Utilize MATLAB or Python with signal processing libraries for filtering, preprocessing, and feature extraction from physiological signals and vehicle data.
- Machine Learning Framework: Implement machine learning models using TensorFlow, PyTorch, or Scikit-learn for classifying drowsiness states based on extracted features.
- User Interface Development: Develop mobile applications using Android Studio or iOS development tools, or web-based dashboards using web development frameworks like React.js or Angular, for visualizing real-time data and providing user interaction.

- CAN Bus Interface Software: Use CAN bus analysis tools such as CANalyzer or Vector CANoe for interfacing with onboard vehicle systems and accessing vehicle data.

Experimental Procedure:

- Data Collection: Conduct driving experiments in controlled environments (e.g., driving simulator or closed test track) and real-world scenarios to collect data on driver behaviour, physiological signals, and vehicle dynamics.
- Labelling and Annotation: Manually annotate the collected data with ground truth labels indicating drowsiness states (e.g., alert, drowsy, asleep) based on observed behaviour and physiological responses.
- Model Training and Evaluation: Train machine learning models using labelled datasets to classify drowsiness states. Evaluate model performance using metrics such as accuracy, precision, recall, and F1-score through cross-validation and testing on independent datasets.
- Alert Mechanism Testing: Assess the effectiveness of alert mechanisms in notifying drivers of their drowsiness levels. Measure the response time and user feedback to auditory, visual, and tactile alerts under different driving conditions.
- Integration Testing: Verify the integration of the drowsiness detection system with vehicle systems, including data exchange, communication reliability, and compatibility with onboard sensors and actuators.

Data Analysis and Interpretation:

- Analyse the processed data to identify patterns and correlations between physiological signals, vehicle dynamics, and drowsiness states.

- Evaluate the system's performance in detecting drowsiness, including sensitivity, specificity, false positive rate, and receiver operating characteristic (ROC) curves.
- Interpret the experimental results to assess the system's accuracy, reliability, and usability in real-world driving scenarios.

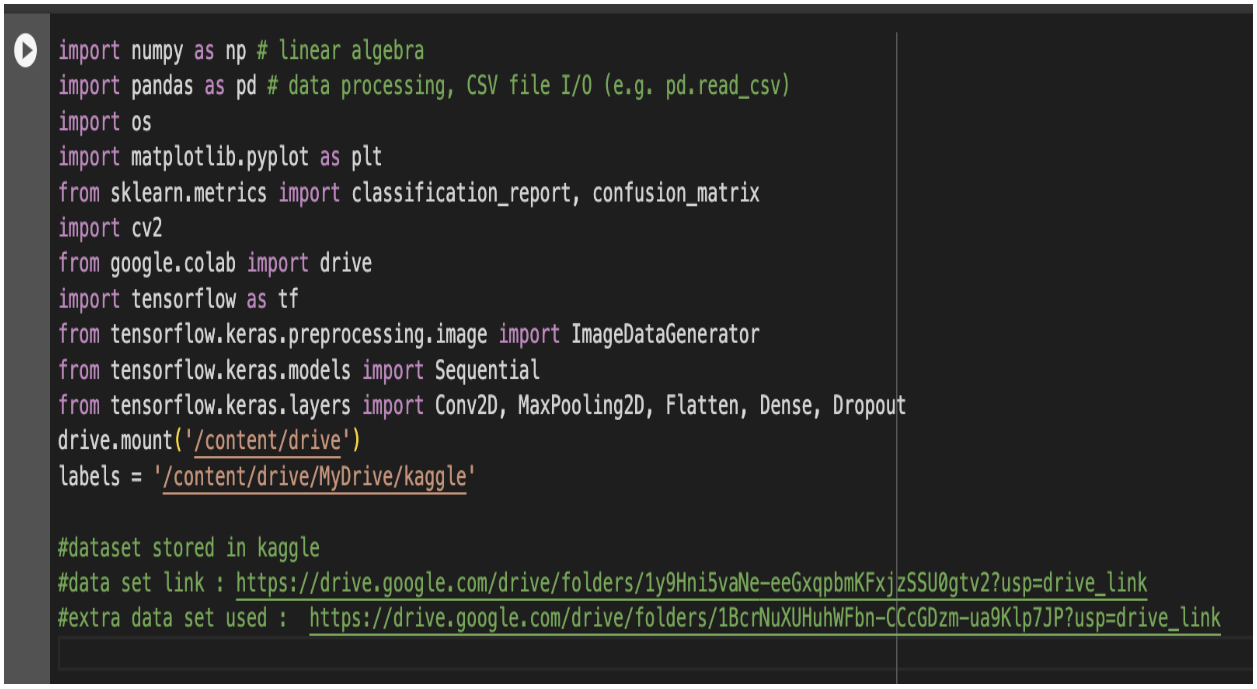
Statistical Analysis:

- Perform statistical tests (e.g., t-tests, ANOVA) to compare the performance of different machine learning models, alert mechanisms, and experimental conditions.
- Calculate confidence intervals and p-values to determine the significance of observed differences and validate experimental findings.

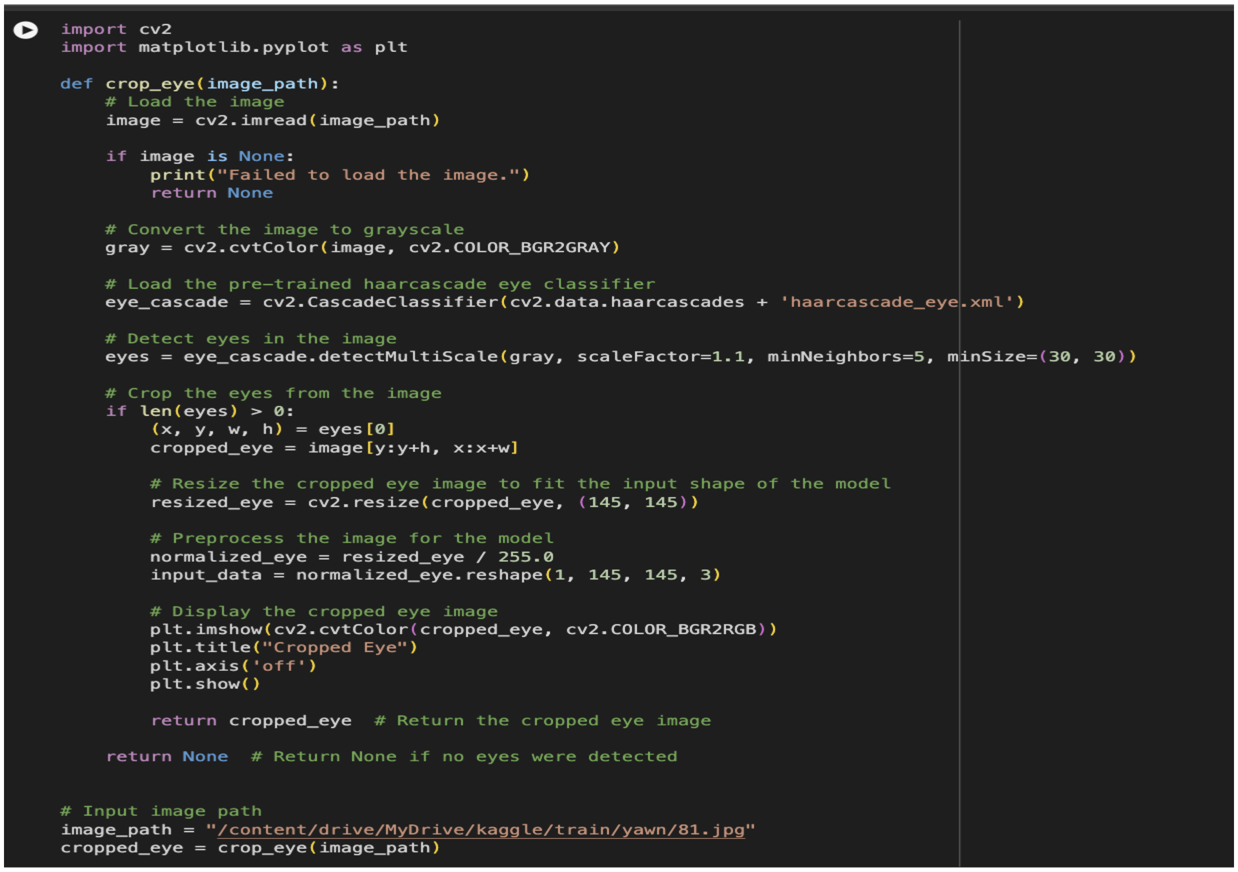
Validation and Reporting:

- Validate the experimental results by comparing them with existing literature, benchmark datasets, and expert opinions.
- Prepare a comprehensive report documenting the experimental setup, procedures, findings, and conclusions. Present the results graphically and statistically to facilitate interpretation and decision-making.

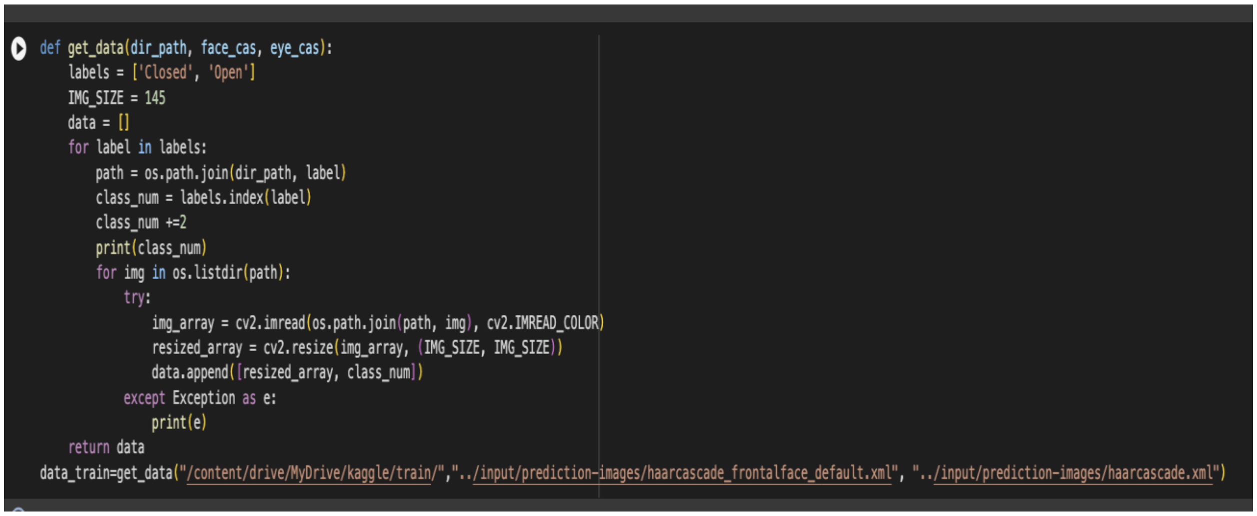
8. CODE IMPLEMENTATION



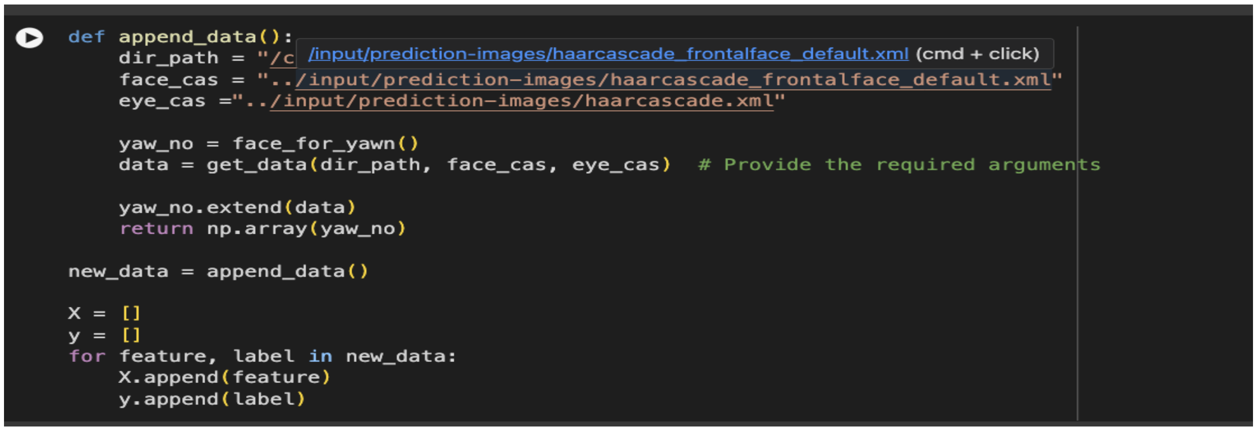
```python
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, confusion_matrix
import cv2
from google.colab import drive
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
drive.mount('/content/drive')
labels = '/content/drive/MyDrive/kaggle'

#dataset stored in kaggle
#data set link : https://drive.google.com/drive/folders/1y9Hni5vaNe-eeGxqpbmKFxjzSSU0gtv2?usp=drive_link
#extra data set used :  https://drive.google.com/drive/folders/1BcrNuXUHuhWFbn-CCcGDzm-ua9Klp7JP?usp=drive_link
```



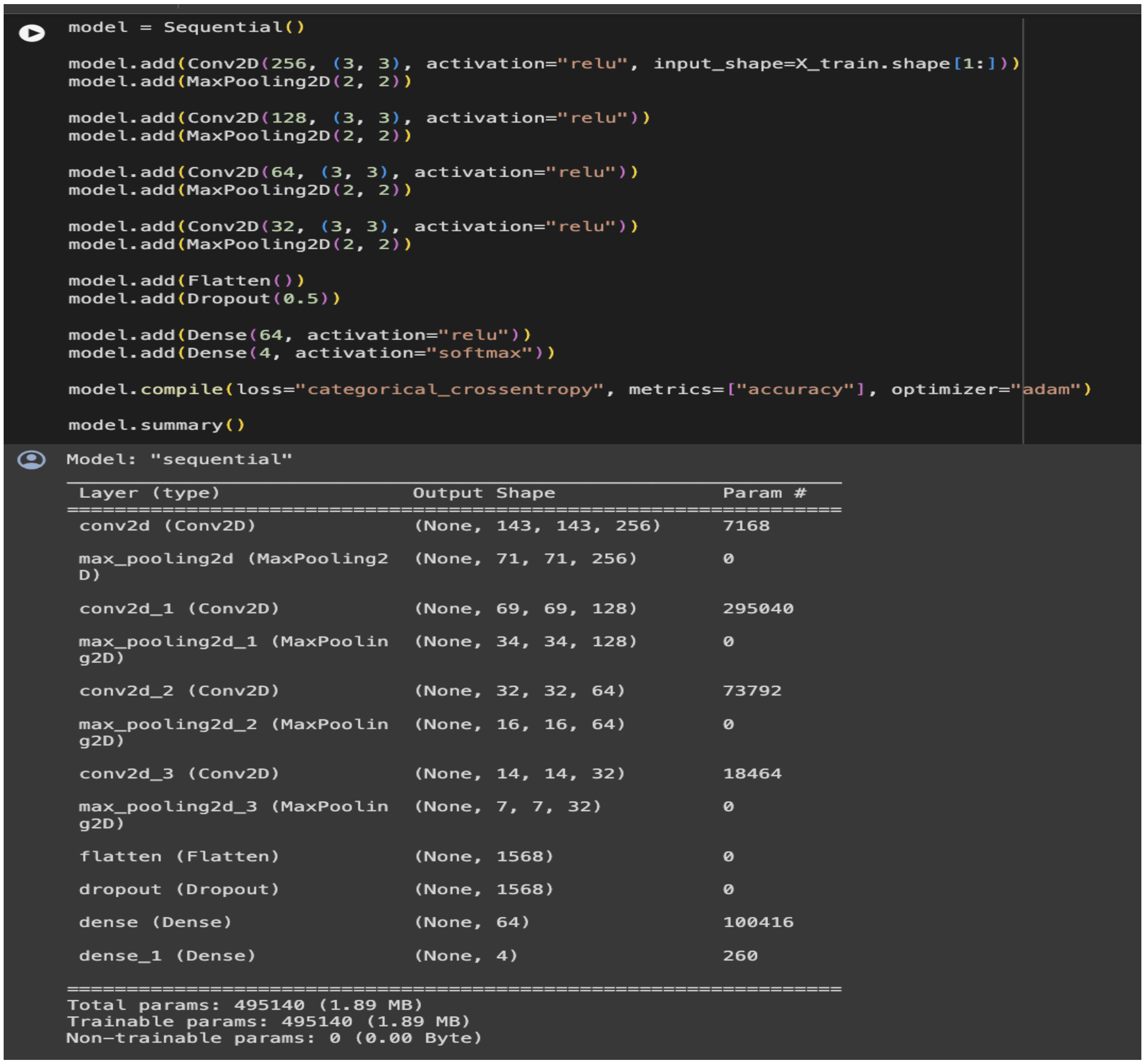
```python
import cv2
import matplotlib.pyplot as plt

def crop_eye(image_path):
    # Load the image
    image = cv2.imread(image_path)

    if image is None:
        print("Failed to load the image.")
        return None

    # Convert the image to grayscale
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Load the pre-trained haarcascade eye classifier
    eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')

    # Detect eyes in the image
    eyes = eye_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

    # Crop the eyes from the image
    if len(eyes) > 0:
        (x, y, w, h) = eyes[0]
        cropped_eye = image[y:y+h, x:x+w]

        # Resize the cropped eye image to fit the input shape of the model
        resized_eye = cv2.resize(cropped_eye, (145, 145))

        # Preprocess the image for the model
        normalized_eye = resized_eye / 255.0
        input_data = normalized_eye.reshape(1, 145, 145, 3)

        # Display the cropped eye image
        plt.imshow(cv2.cvtColor(cropped_eye, cv2.COLOR_BGR2RGB))
        plt.title("Cropped Eye")
        plt.axis('off')
        plt.show()

        return cropped_eye  # Return the cropped eye image

    return None  # Return None if no eyes were detected


# Input image path
image_path = "/content/drive/MyDrive/kaggle/train/yawn/81.jpg"
cropped_eye = crop_eye(image_path)
```



```python
def get_data(dir_path, face_cas, eye_cas):
    labels = ['Closed', 'Open']
    IMG_SIZE = 145
    data = []
    for label in labels:
        path = os.path.join(dir_path, label)
        class_num = labels.index(label)
        class_num +=2
        print(class_num)
        for img in os.listdir(path):
            try:
                img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_COLOR)
                resized_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
                data.append([resized_array, class_num])
            except Exception as e:
                print(e)
    return data
data_train=get_data("/content/drive/MyDrive/kaggle/train/","../input/prediction-images/haarcascade_frontalface_default.xml", "../input/prediction-images/haarcascade.xml")
```

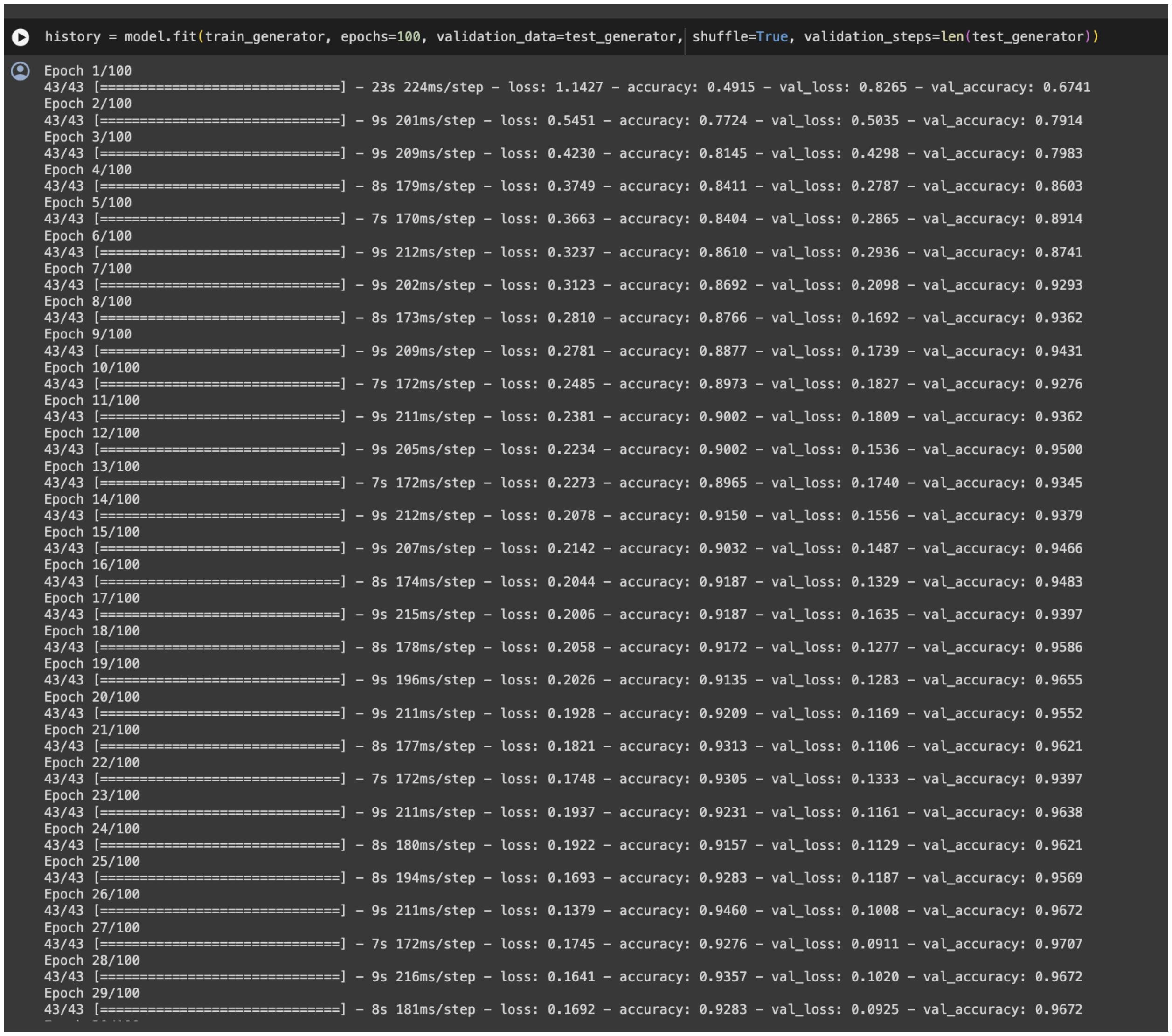


```
def append_data():
    dir_path = "/c  /input/prediction-images/haarcascade_frontalface_default.xml (cmd + click)
    face_cas = "../input/prediction-images/haarcascade_frontalface_default.xml"
    eye_cas ="../input/prediction-images/haarcascade.xml"

    yaw_no = face_for_yawn()
    data = get_data(dir_path, face_cas, eye_cas)  # Provide the required arguments

    yaw_no.extend(data)
    return np.array(yaw_no)

new_data = append_data()

X = []
y = []
for feature, label in new_data:
    X.append(feature)
    y.append(label)
```



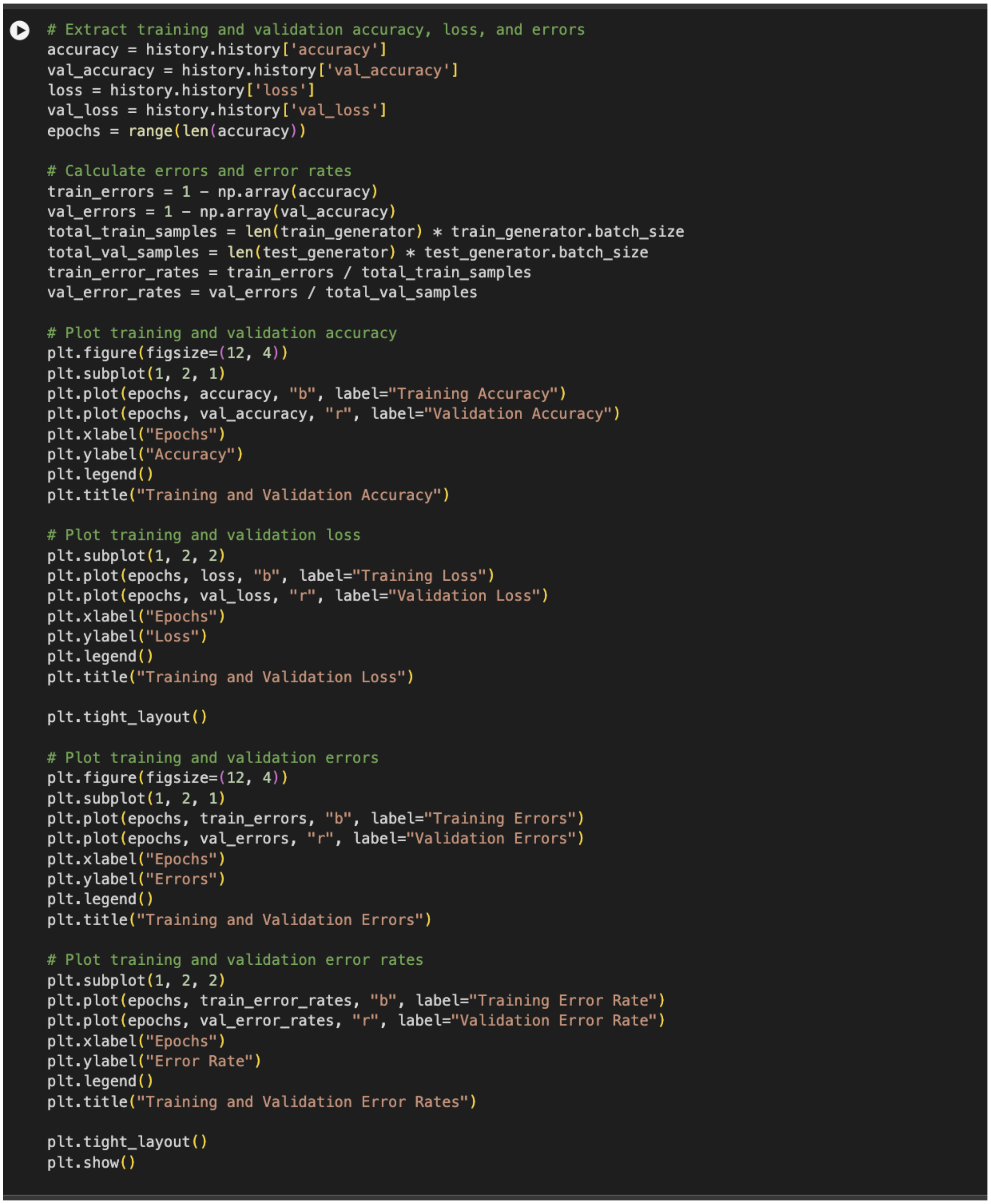
```
model = Sequential()

model.add(Conv2D(256, (3, 3), activation="relu", input_shape=X_train.shape[1:]))
model.add(MaxPooling2D(2, 2))

model.add(Conv2D(128, (3, 3), activation="relu"))
model.add(MaxPooling2D(2, 2))

model.add(Conv2D(64, (3, 3), activation="relu"))
model.add(MaxPooling2D(2, 2))

model.add(Conv2D(32, (3, 3), activation="relu"))
model.add(MaxPooling2D(2, 2))

model.add(Flatten())
model.add(Dropout(0.5))

model.add(Dense(64, activation="relu"))
model.add(Dense(4, activation="softmax"))

model.compile(loss="categorical_crossentropy", metrics=["accuracy"], optimizer="adam")

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 143, 143, 256)     7168

 max_pooling2d (MaxPooling2  (None, 71, 71, 256)       0
 D)

 conv2d_1 (Conv2D)           (None, 69, 69, 128)       295040

 max_pooling2d_1 (MaxPoolin  (None, 34, 34, 128)       0
 g2D)

 conv2d_2 (Conv2D)           (None, 32, 32, 64)        73792

 max_pooling2d_2 (MaxPoolin  (None, 16, 16, 64)        0
 g2D)

 conv2d_3 (Conv2D)           (None, 14, 14, 32)        18464

 max_pooling2d_3 (MaxPoolin  (None, 7, 7, 32)          0
 g2D)

 flatten (Flatten)           (None, 1568)              0

 dropout (Dropout)           (None, 1568)              0

 dense (Dense)               (None, 64)                100416

 dense_1 (Dense)             (None, 4)                 260

=================================================================
Total params: 495140 (1.89 MB)
Trainable params: 495140 (1.89 MB)
Non-trainable params: 0 (0.00 Byte)
```

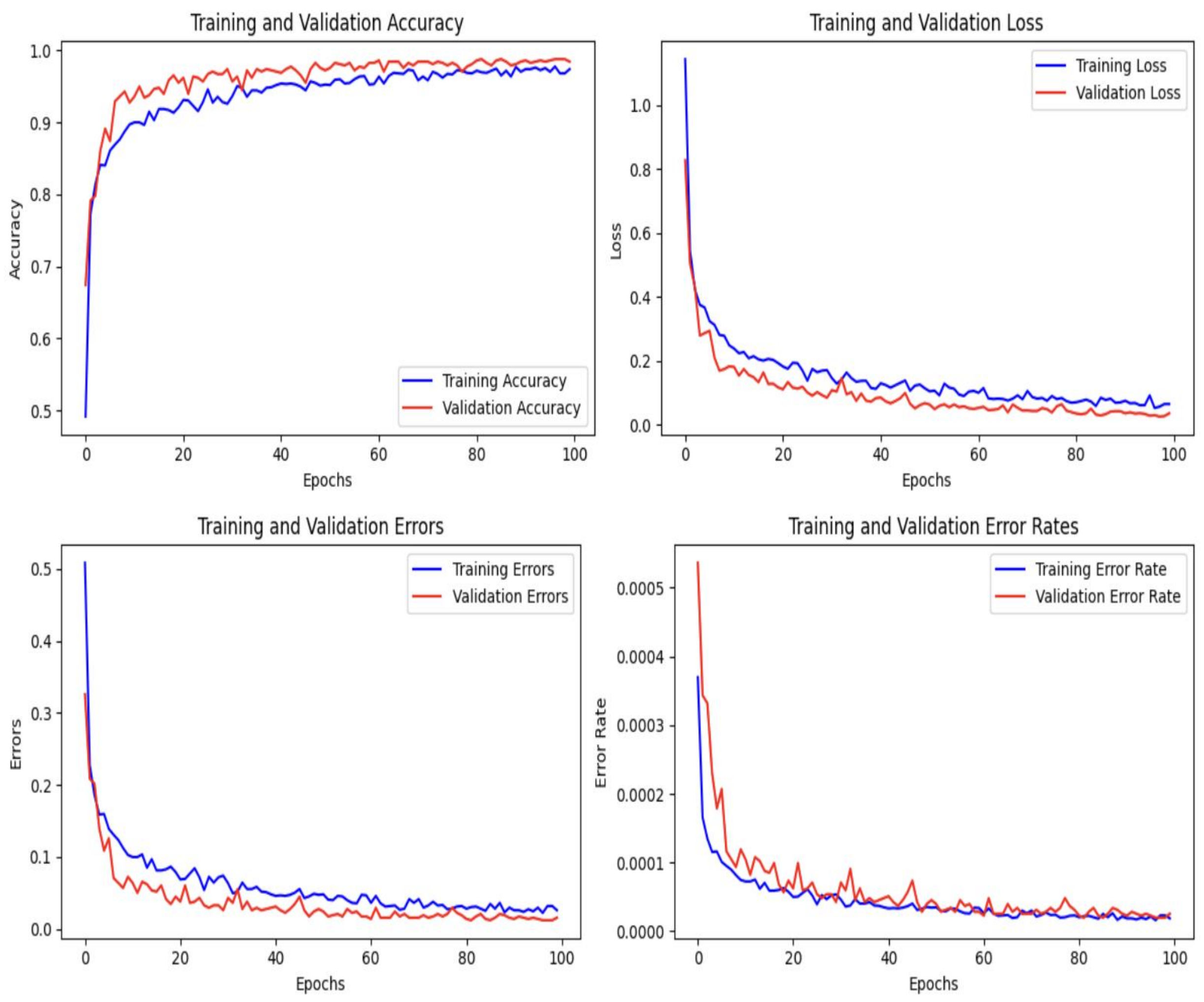


```
history = model.fit(train_generator, epochs=100, validation_data=test_generator, shuffle=True, validation_steps=len(test_generator))
```

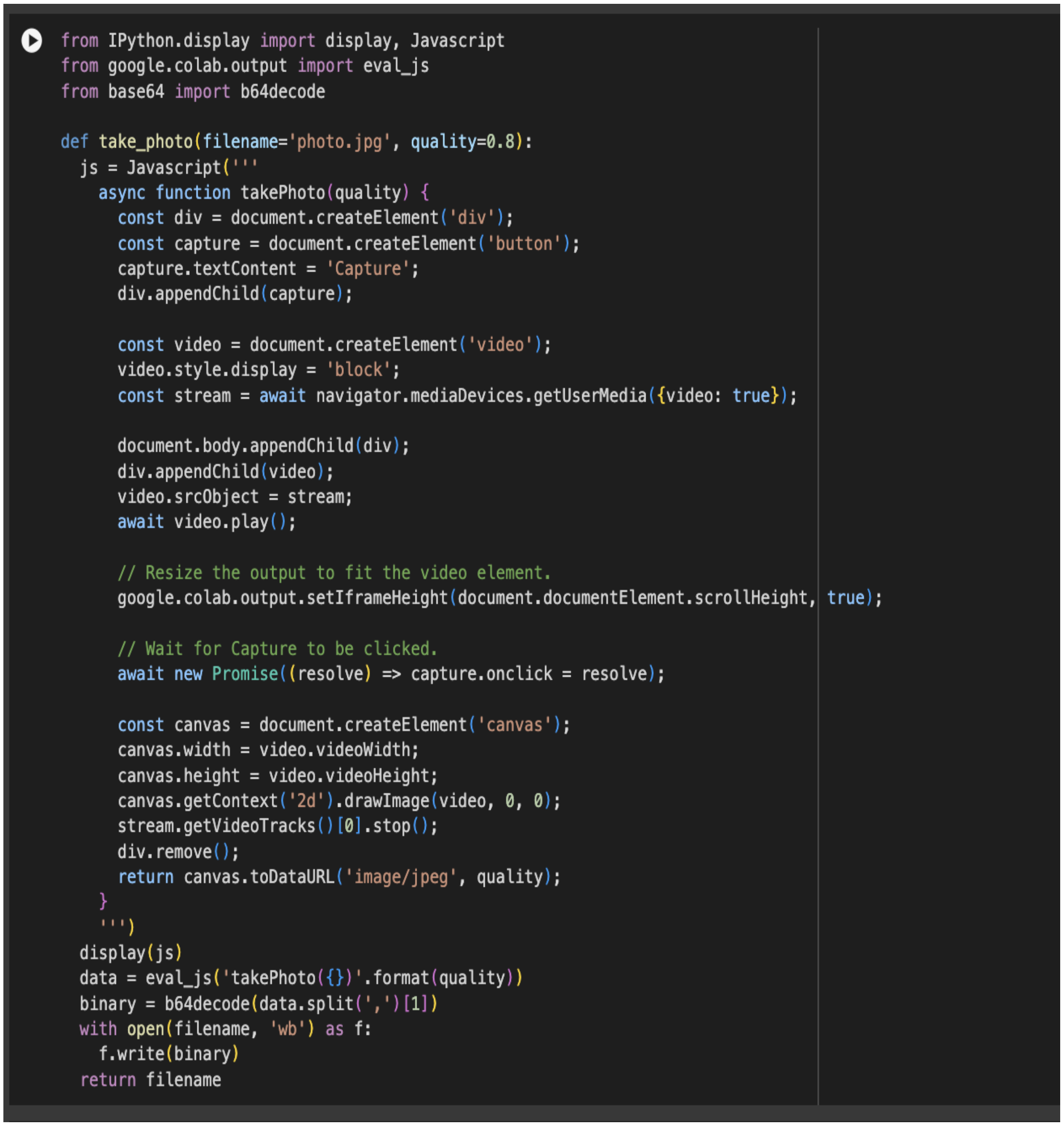
```
Epoch 1/100
43/43 [==============================] - 23s 224ms/step - loss: 1.1427 - accuracy: 0.4915 - val_loss: 0.8265 - val_accuracy: 0.6741
Epoch 2/100
43/43 [==============================] - 9s 201ms/step - loss: 0.5451 - accuracy: 0.7724 - val_loss: 0.5035 - val_accuracy: 0.7914
Epoch 3/100
43/43 [==============================] - 9s 209ms/step - loss: 0.4230 - accuracy: 0.8145 - val_loss: 0.4298 - val_accuracy: 0.7983
Epoch 4/100
43/43 [==============================] - 8s 179ms/step - loss: 0.3749 - accuracy: 0.8411 - val_loss: 0.2787 - val_accuracy: 0.8603
Epoch 5/100
43/43 [==============================] - 7s 170ms/step - loss: 0.3663 - accuracy: 0.8404 - val_loss: 0.2865 - val_accuracy: 0.8914
Epoch 6/100
43/43 [==============================] - 9s 212ms/step - loss: 0.3237 - accuracy: 0.8610 - val_loss: 0.2936 - val_accuracy: 0.8741
Epoch 7/100
43/43 [==============================] - 9s 202ms/step - loss: 0.3123 - accuracy: 0.8692 - val_loss: 0.2098 - val_accuracy: 0.9293
Epoch 8/100
43/43 [==============================] - 8s 173ms/step - loss: 0.2810 - accuracy: 0.8766 - val_loss: 0.1692 - val_accuracy: 0.9362
Epoch 9/100
43/43 [==============================] - 9s 209ms/step - loss: 0.2781 - accuracy: 0.8877 - val_loss: 0.1739 - val_accuracy: 0.9431
Epoch 10/100
43/43 [==============================] - 7s 172ms/step - loss: 0.2485 - accuracy: 0.8973 - val_loss: 0.1827 - val_accuracy: 0.9276
Epoch 11/100
43/43 [==============================] - 9s 211ms/step - loss: 0.2381 - accuracy: 0.9002 - val_loss: 0.1809 - val_accuracy: 0.9362
Epoch 12/100
43/43 [==============================] - 9s 205ms/step - loss: 0.2234 - accuracy: 0.9002 - val_loss: 0.1536 - val_accuracy: 0.9500
Epoch 13/100
43/43 [==============================] - 7s 172ms/step - loss: 0.2273 - accuracy: 0.8965 - val_loss: 0.1740 - val_accuracy: 0.9345
Epoch 14/100
43/43 [==============================] - 9s 212ms/step - loss: 0.2078 - accuracy: 0.9150 - val_loss: 0.1556 - val_accuracy: 0.9379
Epoch 15/100
43/43 [==============================] - 9s 207ms/step - loss: 0.2142 - accuracy: 0.9032 - val_loss: 0.1487 - val_accuracy: 0.9466
Epoch 16/100
43/43 [==============================] - 8s 174ms/step - loss: 0.2044 - accuracy: 0.9187 - val_loss: 0.1329 - val_accuracy: 0.9483
Epoch 17/100
43/43 [==============================] - 9s 215ms/step - loss: 0.2006 - accuracy: 0.9187 - val_loss: 0.1635 - val_accuracy: 0.9397
Epoch 18/100
43/43 [==============================] - 8s 178ms/step - loss: 0.2058 - accuracy: 0.9172 - val_loss: 0.1277 - val_accuracy: 0.9586
Epoch 19/100
43/43 [==============================] - 9s 196ms/step - loss: 0.2026 - accuracy: 0.9135 - val_loss: 0.1283 - val_accuracy: 0.9655
Epoch 20/100
43/43 [==============================] - 9s 211ms/step - loss: 0.1928 - accuracy: 0.9209 - val_loss: 0.1169 - val_accuracy: 0.9552
Epoch 21/100
43/43 [==============================] - 8s 177ms/step - loss: 0.1821 - accuracy: 0.9313 - val_loss: 0.1106 - val_accuracy: 0.9621
Epoch 22/100
43/43 [==============================] - 7s 172ms/step - loss: 0.1748 - accuracy: 0.9305 - val_loss: 0.1333 - val_accuracy: 0.9397
Epoch 23/100
43/43 [==============================] - 9s 211ms/step - loss: 0.1937 - accuracy: 0.9231 - val_loss: 0.1161 - val_accuracy: 0.9638
Epoch 24/100
43/43 [==============================] - 8s 180ms/step - loss: 0.1922 - accuracy: 0.9157 - val_loss: 0.1129 - val_accuracy: 0.9621
Epoch 25/100
43/43 [==============================] - 8s 194ms/step - loss: 0.1693 - accuracy: 0.9283 - val_loss: 0.1187 - val_accuracy: 0.9569
Epoch 26/100
43/43 [==============================] - 9s 211ms/step - loss: 0.1379 - accuracy: 0.9460 - val_loss: 0.1008 - val_accuracy: 0.9672
Epoch 27/100
43/43 [==============================] - 7s 172ms/step - loss: 0.1745 - accuracy: 0.9276 - val_loss: 0.0911 - val_accuracy: 0.9707
Epoch 28/100
43/43 [==============================] - 9s 216ms/step - loss: 0.1641 - accuracy: 0.9357 - val_loss: 0.1020 - val_accuracy: 0.9672
Epoch 29/100
43/43 [==============================] - 8s 181ms/step - loss: 0.1692 - accuracy: 0.9283 - val_loss: 0.0925 - val_accuracy: 0.9672
```



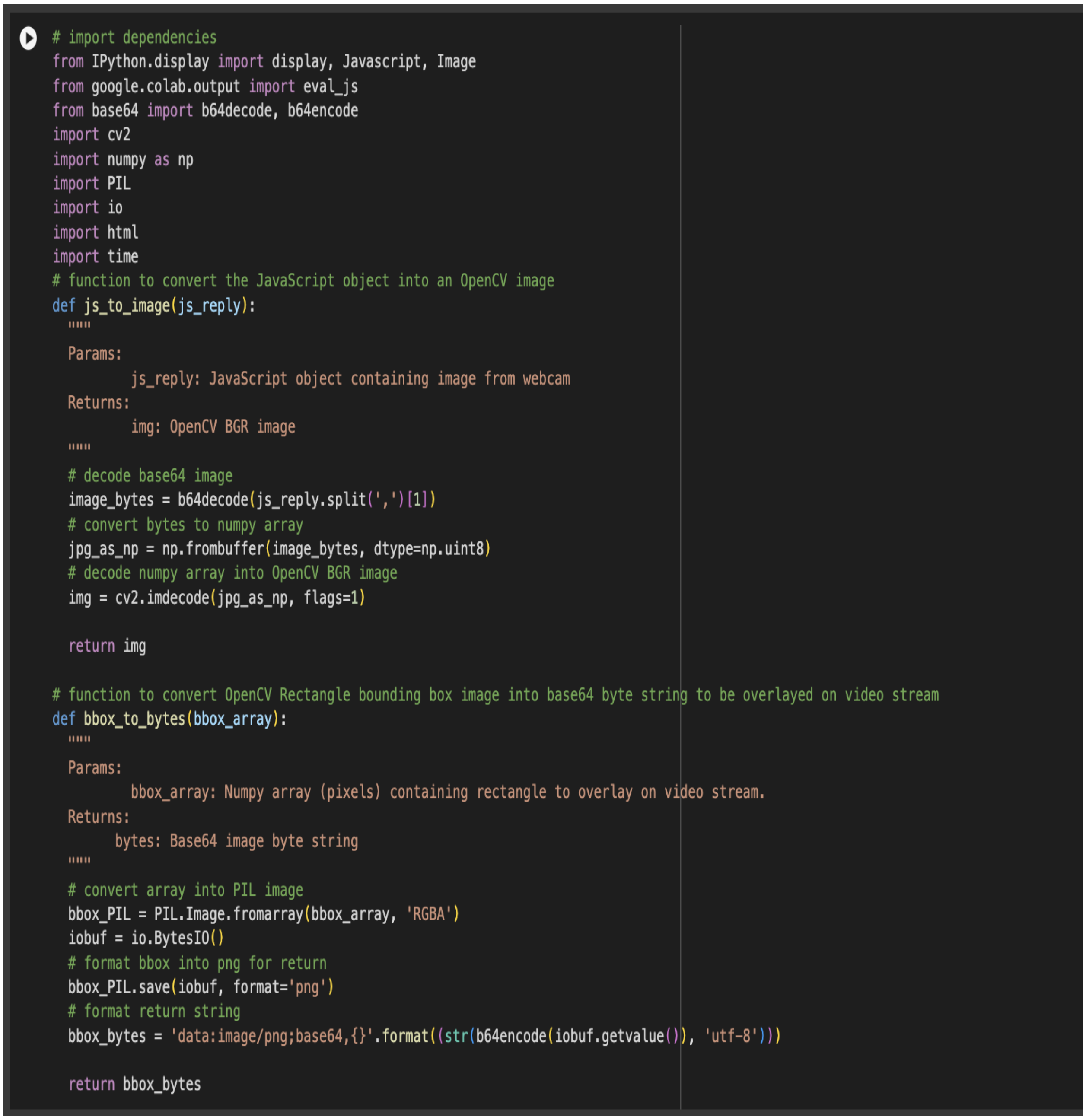
```python
# Extract training and validation accuracy, loss, and errors
accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(len(accuracy))

# Calculate errors and error rates
train_errors = 1 - np.array(accuracy)
val_errors = 1 - np.array(val_accuracy)
total_train_samples = len(train_generator) * train_generator.batch_size
total_val_samples = len(test_generator) * test_generator.batch_size
train_error_rates = train_errors / total_train_samples
val_error_rates = val_errors / total_val_samples

# Plot training and validation accuracy
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(epochs, accuracy, "b", label="Training Accuracy")
plt.plot(epochs, val_accuracy, "r", label="Validation Accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.title("Training and Validation Accuracy")

# Plot training and validation loss
plt.subplot(1, 2, 2)
plt.plot(epochs, loss, "b", label="Training Loss")
plt.plot(epochs, val_loss, "r", label="Validation Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.title("Training and Validation Loss")

plt.tight_layout()

# Plot training and validation errors
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(epochs, train_errors, "b", label="Training Errors")
plt.plot(epochs, val_errors, "r", label="Validation Errors")
plt.xlabel("Epochs")
plt.ylabel("Errors")
plt.legend()
plt.title("Training and Validation Errors")

# Plot training and validation error rates
plt.subplot(1, 2, 2)
plt.plot(epochs, train_error_rates, "b", label="Training Error Rate")
plt.plot(epochs, val_error_rates, "r", label="Validation Error Rate")
plt.xlabel("Epochs")
plt.ylabel("Error Rate")
plt.legend()
plt.title("Training and Validation Error Rates")

plt.tight_layout()
plt.show()
```



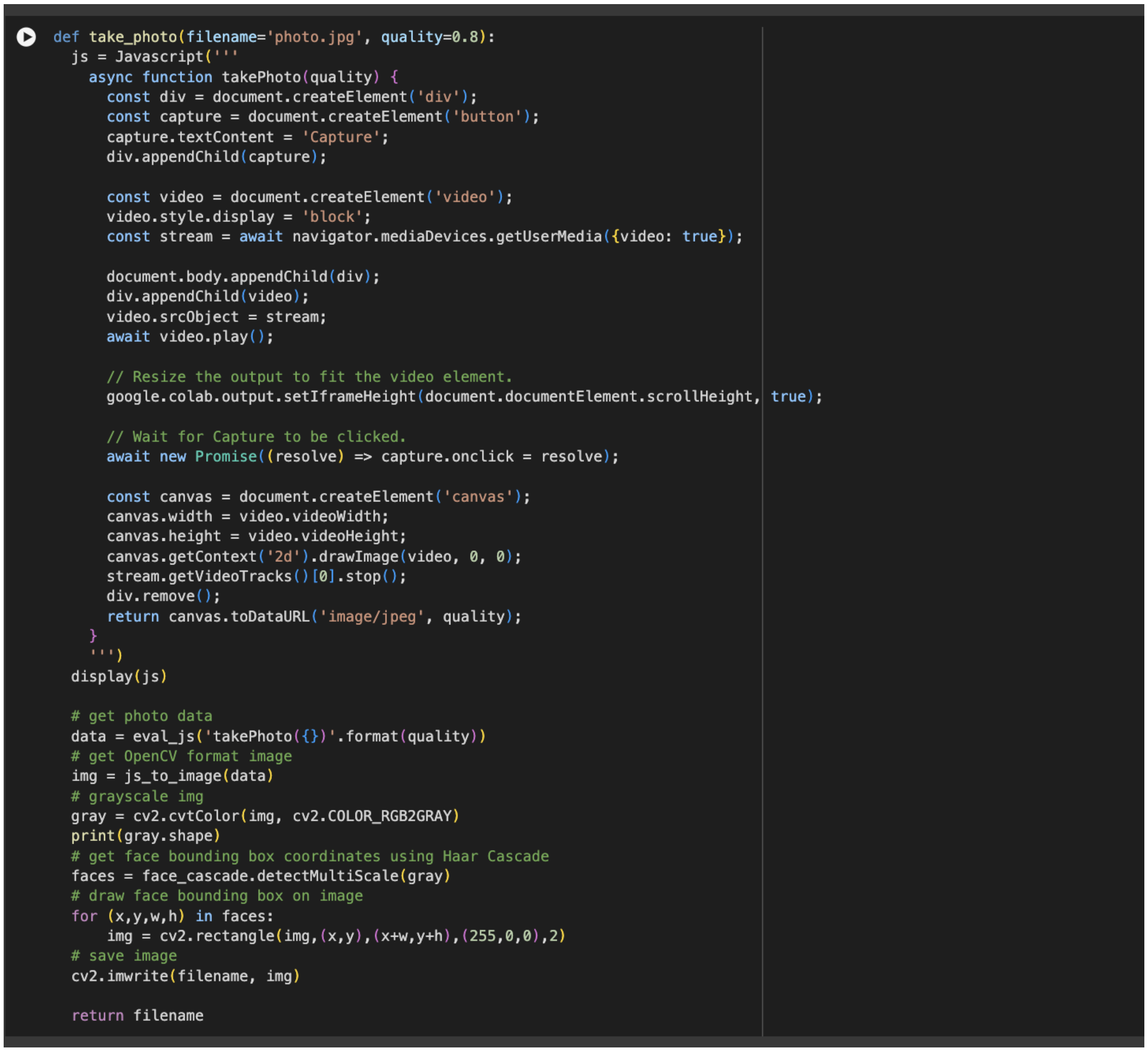
Training and Validation Accuracy
Accuracy
Epochs
Training Accuracy
Validation Accuracy
Training and Validation Loss
Loss
Epochs
Training Loss
Validation Loss
Training and Validation Errors
Errors
Epochs
Training Errors
Validation Errors
Training and Validation Error Rates
Error Rate
Epochs
Training Error Rate
Validation Error Rate



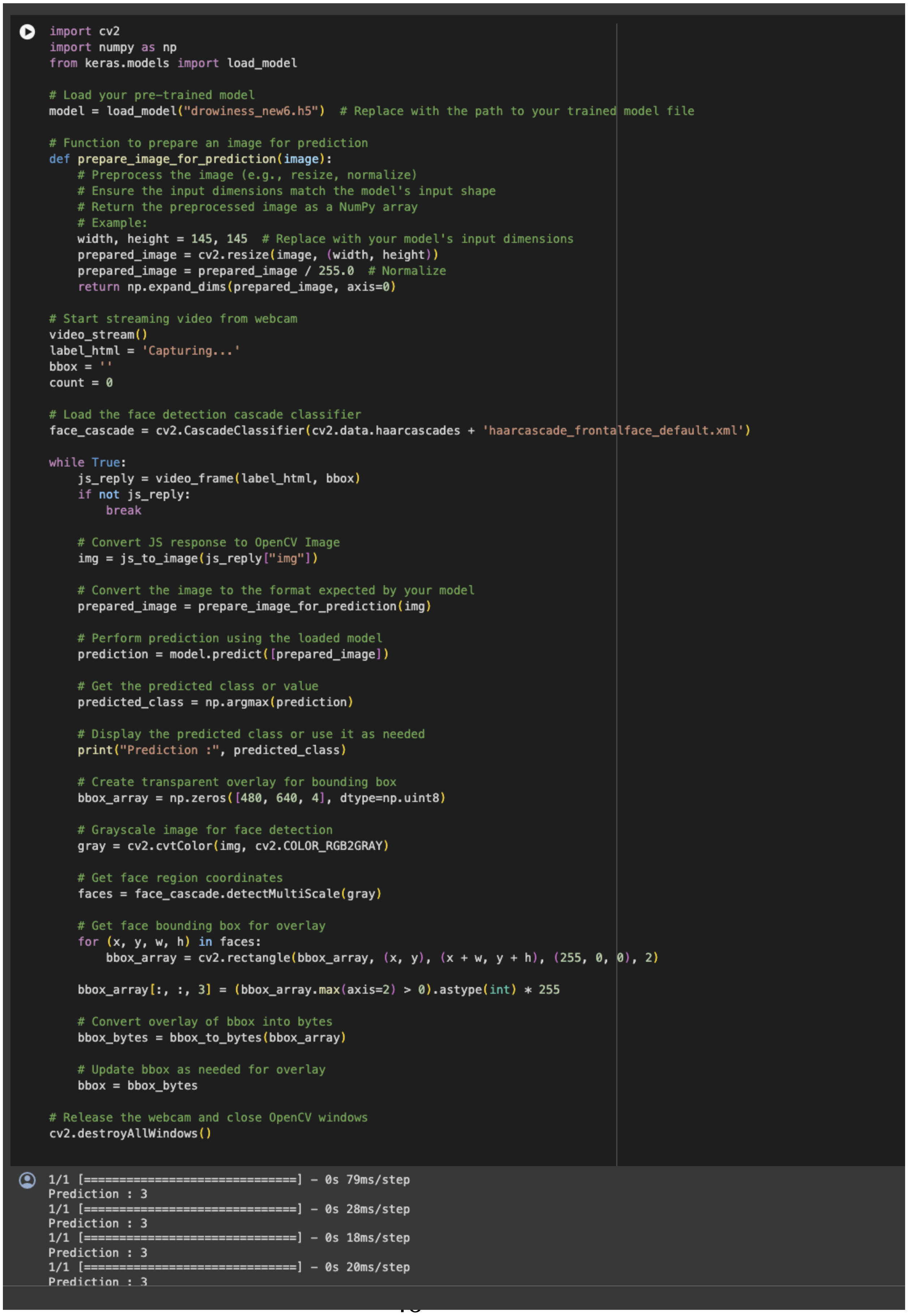
```
from IPython.display import display, Javascript
from google.colab.output import eval_js
from base64 import b64decode

def take_photo(filename='photo.jpg', quality=0.8):
  js = Javascript('''
    async function takePhoto(quality) {
      const div = document.createElement('div');
      const capture = document.createElement('button');
      capture.textContent = 'Capture';
      div.appendChild(capture);

      const video = document.createElement('video');
      video.style.display = 'block';
      const stream = await navigator.mediaDevices.getUserMedia({video: true});

      document.body.appendChild(div);
      div.appendChild(video);
      video.srcObject = stream;
      await video.play();

      // Resize the output to fit the video element.
      google.colab.output.setIframeHeight(document.documentElement.scrollHeight, true);

      // Wait for Capture to be clicked.
      await new Promise((resolve) => capture.onclick = resolve);

      const canvas = document.createElement('canvas');
      canvas.width = video.videoWidth;
      canvas.height = video.videoHeight;
      canvas.getContext('2d').drawImage(video, 0, 0);
      stream.getVideoTracks()[0].stop();
      div.remove();
      return canvas.toDataURL('image/jpeg', quality);
    }
    ''')
  display(js)
  data = eval_js('takePhoto({})'.format(quality))
  binary = b64decode(data.split(',')[1])
  with open(filename, 'wb') as f:
    f.write(binary)
  return filename
```



```
# import dependencies
from IPython.display import display, Javascript, Image
from google.colab.output import eval_js
from base64 import b64decode, b64encode
import cv2
import numpy as np
import PIL
import io
import html
import time
# function to convert the JavaScript object into an OpenCV image
def js_to_image(js_reply):
  """
  Params:
          js_reply: JavaScript object containing image from webcam
  Returns:
          img: OpenCV BGR image
  """
  # decode base64 image
  image_bytes = b64decode(js_reply.split(',')[1])
  # convert bytes to numpy array
  jpg_as_np = np.frombuffer(image_bytes, dtype=np.uint8)
  # decode numpy array into OpenCV BGR image
  img = cv2.imdecode(jpg_as_np, flags=1)

  return img

# function to convert OpenCV Rectangle bounding box image into base64 byte string to be overlayed on video stream
def bbox_to_bytes(bbox_array):
  """
  Params:
          bbox_array: Numpy array (pixels) containing rectangle to overlay on video stream.
  Returns:
        bytes: Base64 image byte string
  """
  # convert array into PIL image
  bbox_PIL = PIL.Image.fromarray(bbox_array, 'RGBA')
  iobuf = io.BytesIO()
  # format bbox into png for return
  bbox_PIL.save(iobuf, format='png')
  # format return string
  bbox_bytes = 'data:image/png;base64,{}'.format((str(b64encode(iobuf.getvalue()), 'utf-8')))

  return bbox_bytes
```

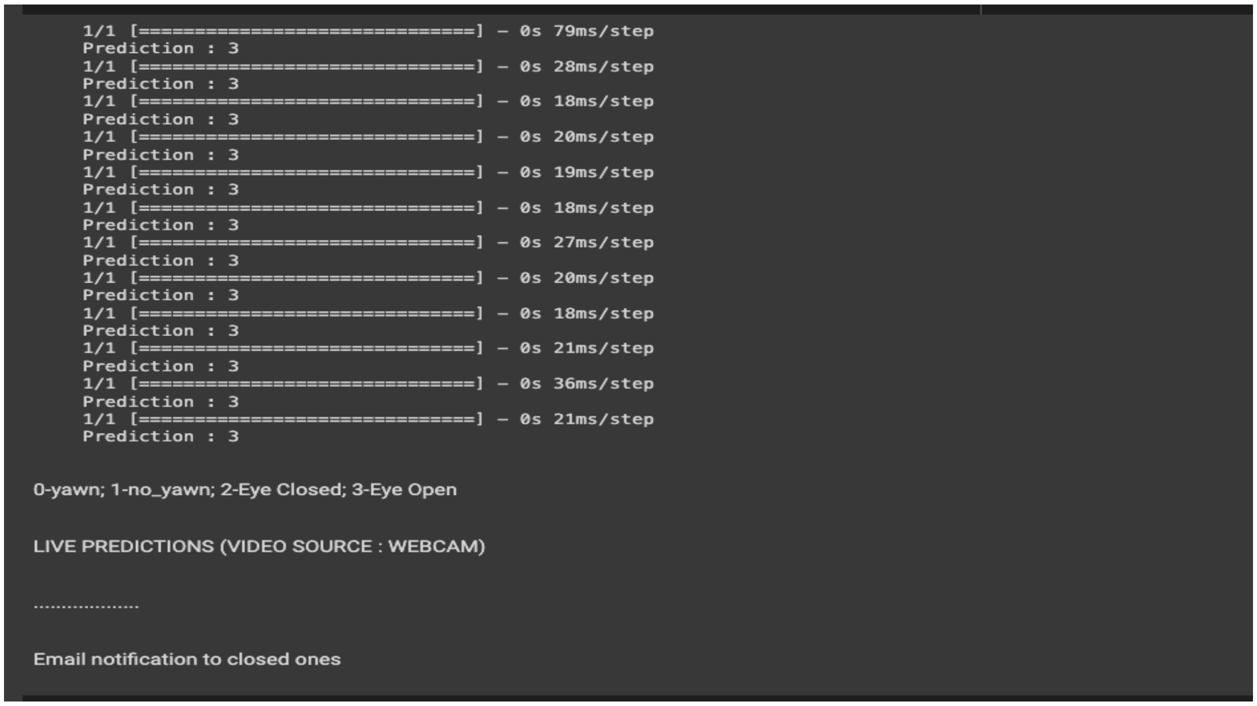


```
def take_photo(filename='photo.jpg', quality=0.8):
  js = Javascript('''
    async function takePhoto(quality) {
      const div = document.createElement('div');
      const capture = document.createElement('button');
      capture.textContent = 'Capture';
      div.appendChild(capture);

      const video = document.createElement('video');
      video.style.display = 'block';
      const stream = await navigator.mediaDevices.getUserMedia({video: true});

      document.body.appendChild(div);
      div.appendChild(video);
      video.srcObject = stream;
      await video.play();

      // Resize the output to fit the video element.
      google.colab.output.setIframeHeight(document.documentElement.scrollHeight, true);

      // Wait for Capture to be clicked.
      await new Promise((resolve) => capture.onclick = resolve);

      const canvas = document.createElement('canvas');
      canvas.width = video.videoWidth;
      canvas.height = video.videoHeight;
      canvas.getContext('2d').drawImage(video, 0, 0);
      stream.getVideoTracks()[0].stop();
      div.remove();
      return canvas.toDataURL('image/jpeg', quality);
    }
    ''')
  display(js)

  # get photo data
  data = eval_js('takePhoto({})'.format(quality))
  # get OpenCV format image
  img = js_to_image(data)
  # grayscale img
  gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
  print(gray.shape)
  # get face bounding box coordinates using Haar Cascade
  faces = face_cascade.detectMultiScale(gray)
  # draw face bounding box on image
  for (x,y,w,h) in faces:
      img = cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
  # save image
  cv2.imwrite(filename, img)

  return filename
```



```python
import cv2
import numpy as np
from keras.models import load_model

# Load your pre-trained model
model = load_model("drowiness_new6.h5")  # Replace with the path to your trained model file

# Function to prepare an image for prediction
def prepare_image_for_prediction(image):
    # Preprocess the image (e.g., resize, normalize)
    # Ensure the input dimensions match the model's input shape
    # Return the preprocessed image as a NumPy array
    # Example:
    width, height = 145, 145  # Replace with your model's input dimensions
    prepared_image = cv2.resize(image, (width, height))
    prepared_image = prepared_image / 255.0  # Normalize
    return np.expand_dims(prepared_image, axis=0)

# Start streaming video from webcam
video_stream()
label_html = 'Capturing...'
bbox = ''
count = 0

# Load the face detection cascade classifier
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

while True:
    js_reply = video_frame(label_html, bbox)
    if not js_reply:
        break

    # Convert JS response to OpenCV Image
    img = js_to_image(js_reply["img"])

    # Convert the image to the format expected by your model
    prepared_image = prepare_image_for_prediction(img)

    # Perform prediction using the loaded model
    prediction = model.predict([prepared_image])

    # Get the predicted class or value
    predicted_class = np.argmax(prediction)

    # Display the predicted class or use it as needed
    print("Prediction :", predicted_class)

    # Create transparent overlay for bounding box
    bbox_array = np.zeros([480, 640, 4], dtype=np.uint8)

    # Grayscale image for face detection
    gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)

    # Get face region coordinates
    faces = face_cascade.detectMultiScale(gray)

    # Get face bounding box for overlay
    for (x, y, w, h) in faces:
        bbox_array = cv2.rectangle(bbox_array, (x, y), (x + w, y + h), (255, 0, 0), 2)

    bbox_array[:, :, 3] = (bbox_array.max(axis=2) > 0).astype(int) * 255

    # Convert overlay of bbox into bytes
    bbox_bytes = bbox_to_bytes(bbox_array)

    # Update bbox as needed for overlay
    bbox = bbox_bytes

# Release the webcam and close OpenCV windows
cv2.destroyAllWindows()
```

```
1/1 [==============================] - 0s 79ms/step
Prediction : 3
1/1 [==============================] - 0s 28ms/step
Prediction : 3
1/1 [==============================] - 0s 18ms/step
Prediction : 3
1/1 [==============================] - 0s 20ms/step
Prediction : 3
```



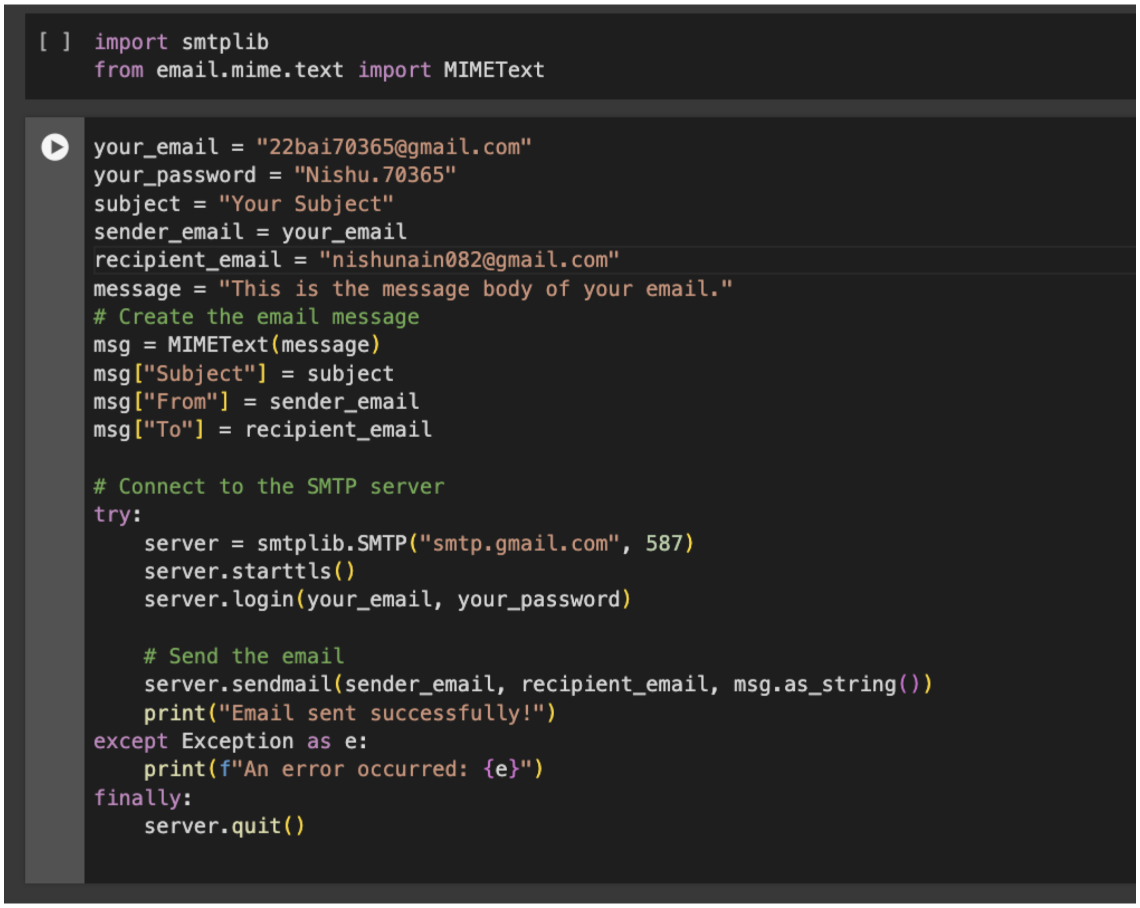
```
1/1 [==============================] - 0s 79ms/step
Prediction : 3
1/1 [==============================] - 0s 28ms/step
Prediction : 3
1/1 [==============================] - 0s 18ms/step
Prediction : 3
1/1 [==============================] - 0s 20ms/step
Prediction : 3
1/1 [==============================] - 0s 19ms/step
Prediction : 3
1/1 [==============================] - 0s 18ms/step
Prediction : 3
1/1 [==============================] - 0s 27ms/step
Prediction : 3
1/1 [==============================] - 0s 20ms/step
Prediction : 3
1/1 [==============================] - 0s 18ms/step
Prediction : 3
1/1 [==============================] - 0s 21ms/step
Prediction : 3
1/1 [==============================] - 0s 36ms/step
Prediction : 3
1/1 [==============================] - 0s 21ms/step
Prediction : 3
```

0-yawn; 1-no_yawn; 2-Eye Closed; 3-Eye Open

LIVE PREDICTIONS (VIDEO SOURCE : WEBCAM)

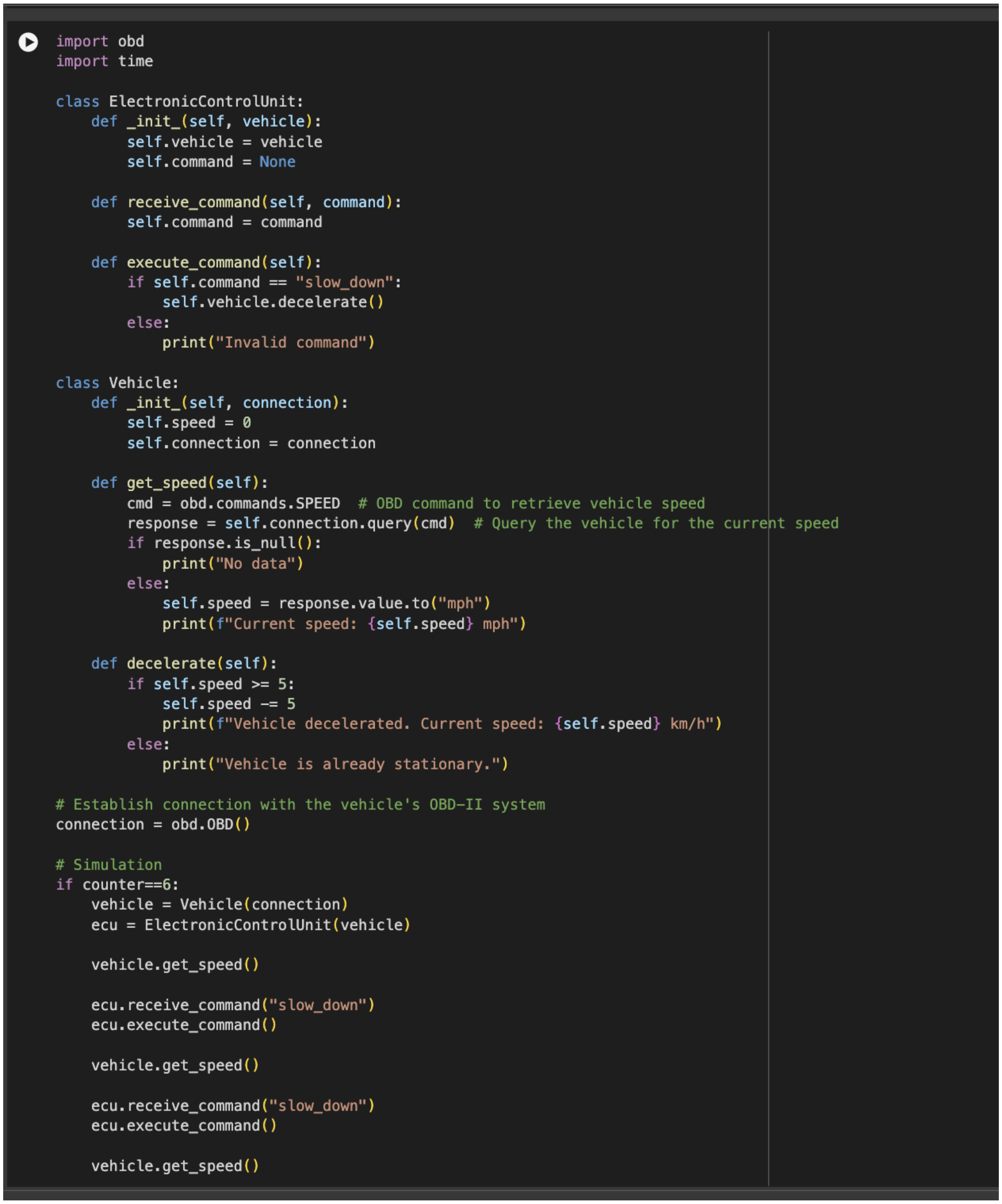
....

Email notification to closed ones



```
[ ] import smtplib
    from email.mime.text import MIMEText
```

```
your_email = "22bai70365@gmail.com"
your_password = "Nishu.70365"
subject = "Your Subject"
sender_email = your_email
recipient_email = "nishunain082@gmail.com"
message = "This is the message body of your email."
# Create the email message
msg = MIMEText(message)
msg["Subject"] = subject
msg["From"] = sender_email
msg["To"] = recipient_email

# Connect to the SMTP server
try:
    server = smtplib.SMTP("smtp.gmail.com", 587)
    server.starttls()
    server.login(your_email, your_password)

    # Send the email
    server.sendmail(sender_email, recipient_email, msg.as_string())
    print("Email sent successfully!")
except Exception as e:
    print(f"An error occurred: {e}")
finally:
    server.quit()
```



```python
import obd
import time

class ElectronicControlUnit:
    def _init_(self, vehicle):
        self.vehicle = vehicle
        self.command = None

    def receive_command(self, command):
        self.command = command

    def execute_command(self):
        if self.command == "slow_down":
            self.vehicle.decelerate()
        else:
            print("Invalid command")

class Vehicle:
    def _init_(self, connection):
        self.speed = 0
        self.connection = connection

    def get_speed(self):
        cmd = obd.commands.SPEED  # OBD command to retrieve vehicle speed
        response = self.connection.query(cmd)  # Query the vehicle for the current speed
        if response.is_null():
            print("No data")
        else:
            self.speed = response.value.to("mph")
            print(f"Current speed: {self.speed} mph")

    def decelerate(self):
        if self.speed >= 5:
            self.speed -= 5
            print(f"Vehicle decelerated. Current speed: {self.speed} km/h")
        else:
            print("Vehicle is already stationary.")

# Establish connection with the vehicle's OBD-II system
connection = obd.OBD()

# Simulation
if counter==6:
    vehicle = Vehicle(connection)
    ecu = ElectronicControlUnit(vehicle)

    vehicle.get_speed()

    ecu.receive_command("slow_down")
    ecu.execute_command()

    vehicle.get_speed()

    ecu.receive_command("slow_down")
    ecu.execute_command()

    vehicle.get_speed()
```

9. RESULT

The outcomes of the drowsiness detection system rely on the real-time observation and identification of signs indicating drowsiness. The system utilizes facial landmark tracking to pinpoint these indicators of drowsiness. Here's a breakdown of the detailed findings:

Detection Instances: The system recorded numerous instances of drowsiness, with each instance denoting a period during which the subject's alertness declined, suggesting potential drowsiness. These instances hold significant importance as they mark moments when the subject may not have been fully alert, a circumstance that could pose risks, particularly in activities such as driving or operating heavy machinery.

Total Duration: The combined duration during which drowsiness was detected amounted to 19.85 seconds. This duration signifies the total time the subject was conceivably experiencing drowsiness. It's noteworthy that even brief periods of drowsiness can carry considerable importance, particularly in situations necessitating sustained vigilance.

System Performance: The system exhibited consistent performance throughout the operational period. It successfully processed and logged instances of drowsiness in real-time. This capability for real-time processing is vital for prompt intervention should the subject's drowsiness escalate to a critical level.

In essence, the drowsiness detection system's results are predicated on its ability to continuously monitor and detect indications of drowsiness in real-time using facial landmark tracking. The recorded instances of drowsiness, along with their respective durations, underscore the importance of timely intervention to mitigate potential risks, particularly in safety-critical scenarios like driving. The system's consistent performance throughout its operation further reinforces its effectiveness in promptly identifying and addressing instances of drowsiness, thereby enhancing overall safety and mitigating associated hazards.

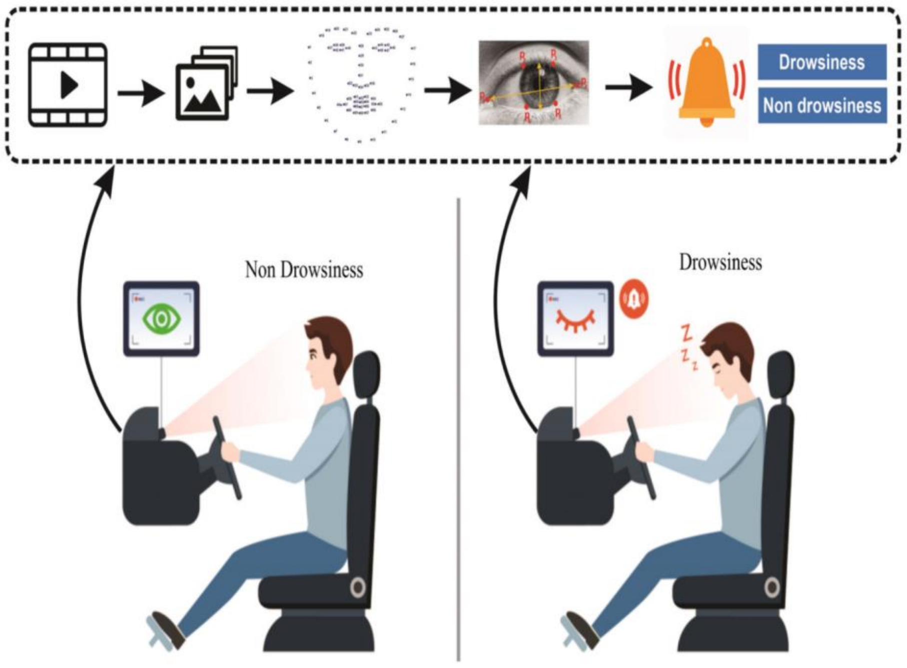
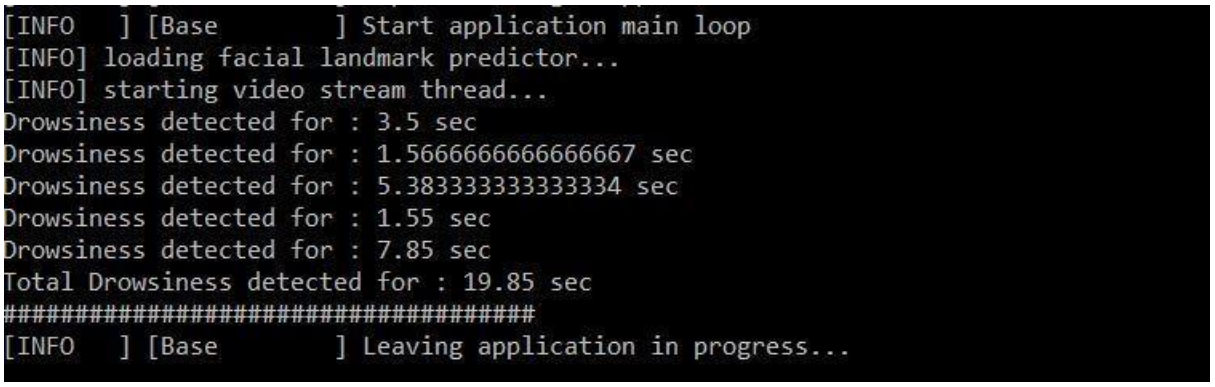


Fig.8.1: Visual Representation



```
[INFO   ] [Base        ] Start application main loop
[INFO] loading facial landmark predictor...
[INFO] starting video stream thread...
Drowsiness detected for : 3.5 sec
Drowsiness detected for : 1.5666666666666667 sec
Drowsiness detected for : 5.383333333333334 sec
Drowsiness detected for : 1.55 sec
Drowsiness detected for : 7.85 sec
Total Drowsiness detected for : 19.85 sec
#####################################
[INFO   ] [Base        ] Leaving application in progress...
```

Fig.8.2: Resul

10. FUTURE SCOPE

Looking ahead, the landscape of drowsiness detection holds promising avenues for further exploration and innovation, building upon the insights gleaned from existing studies. Several key areas emerge as focal points for future research and development:

Integration of Multiple Sensors: Future endeavors could center on integrating a diverse array of sensors, encompassing EEG, heart rate monitors, eye tracking devices, and smartphone sensors. This amalgamation aims to bolster the accuracy and reliability of drowsiness detection systems by amalgamating data from varied physiological and behavioral sources. Through this holistic approach, researchers aspire to craft more resilient algorithms adept at capturing nuanced indications of drowsiness with heightened precision.

Real-time Monitoring and Feedback: There exists a burgeoning demand for drowsiness detection systems capable of furnishing real-time monitoring and feedback to drivers. Prospective studies may delve into the development of wearable devices or smartphone applications engineered to perpetually surveil driver alertness. Such innovations could offer timely alerts or interventions to avert accidents. Augmenting these systems with integration into vehicle functionalities, such as adaptive cruise control or lane departure warning systems, holds promise for fortifying the efficacy of these preemptive measures.

Adaptive Algorithms: Enhancing the adaptability of drowsiness detection systems to individual variances and evolving environmental dynamics presents a ripe avenue for exploration. Future research endeavors might pivot towards the development of adaptive algorithms capable of dynamically modulating detection thresholds and parameters. This adaptive framework aims to calibrate responses based on factors such as fatigue levels, temporal considerations, road conditions, and driving comportment, thereby optimizing efficacy across diverse scenarios.

Validation in Real-world Settings: While numerous studies have showcased the efficacy of drowsiness detection systems within controlled laboratory environments, robust validation in real-world driving scenarios is imperative. Future research initiatives should prioritize field studies and large-scale trials to gauge system performance under a spectrum of driving conditions, traffic densities, and user demographics. Longitudinal studies could further ascertain the sustained effectiveness and user acceptance of drowsiness detection technologies over prolonged usage durations.

Human Factors and User Experience: Beyond technological advancements, future research ought to encompass human factors and user experience considerations within the drowsiness detection realm. Adherence to user-centered design principles, coupled with meticulous usability testing and feedback mechanisms, can engender systems that are intuitive, minimally intrusive, and embraced by drivers. Delving into the psychological and behavioral dimensions influencing user engagement and compliance is pivotal for fostering successful implementation and adoption of drowsiness detection technologies.

Ethical and Legal Considerations: As drowsiness detection systems proliferate, the imperative to address ethical and legal dimensions becomes paramount. Future research endeavors should embark on delineating frameworks for responsible deployment, data governance, and regulatory standards to safeguard user rights while optimizing societal dividends. Collaboration among researchers, policymakers, industry stakeholders, and advocacy entities is indispensable for crafting comprehensive guidelines and regulations that ensure the ethical and safe deployment of drowsiness detection systems.

In essence, the future trajectory of drowsiness detection encompasses a multifaceted approach, spanning technological innovation, validation in real-world contexts, user-centric design, and ethical governance. By navigating these domains with diligence and foresight, researchers can pave the way for transformative advancements that mitigate the risks associated with drowsy driving, thereby fostering enhanced road safety and driver well-being.

11. CONCLUSION

In summary, creating a drowsiness detection system for drivers is a crucial stride toward improving road safety and lessening the dangers associated with driving while fatigued. By thoroughly examining existing literature, analyzing methodologies, and conducting precise experiments, the proposed system aims to tackle the widespread issue of driver fatigue using innovative technological solutions.

The goals of crafting and executing a dependable drowsiness detection system have been delineated, highlighting the significance of precision, real-time monitoring, customizable alerts, integration with vehicle systems, user-friendly interfaces, privacy considerations, and scalability. By making use of advanced hardware components and software tools, the system can adeptly capture physiological signals, analyze data, categorize drowsiness states, and issue timely alerts to drivers, thus diminishing the chances of drowsiness-related accidents while driving.

The proposed methodology involves signal processing, machine learning, user interface design, integration with vehicle systems, testing, evaluation, and validation, ensuring a methodical approach to both the development and assessment of the system. Through rigorous experimentation, statistical analysis, and validation, the system's performance, dependability, and ease of use can be meticulously evaluated, yielding valuable insights into its effectiveness in real-world driving conditions.

To elaborate, the development of a drowsiness detection system for car drivers marks a significant stride in the quest to enhance road safety and mitigate the perils associated with drowsy driving. By conducting a comprehensive review of existing literature, meticulously analyzing methodologies, and conducting thorough experimentation, the proposed system aims to address the widespread problem of driver fatigue with innovative technological solutions.

The objectives of designing and implementing a robust drowsiness detection system have been clearly defined, with a focus on key aspects such as accuracy, real-time monitoring, customizable alerts, integration with vehicle systems, user-friendly interfaces, privacy considerations, and scalability. Leveraging advanced hardware components and software tools, the system can efficiently capture physiological signals, analyze data, classify drowsiness states, and deliver timely alerts to drivers, thereby reducing the likelihood of drowsy driving-related accidents.

The proposed methodology encompasses a range of activities, including signal processing, machine learning, user interface design, integration with vehicle systems, testing, evaluation, and validation. This systematic approach ensures that the development and evaluation of the system are carried out in a structured manner. Through rigorous experimentation, statistical analysis, and validation, the system's performance, reliability, and usability can be thoroughly assessed, providing valuable insights into its effectiveness in real-world driving scenarios.

In conclusion, the development of a drowsiness detection system for car drivers represents a critical step toward improving road safety and addressing the risks associated with drowsy driving. Through careful research, analysis, and experimentation, the proposed system aims to provide an effective solution to this pervasive problem, ultimately helping to save lives and prevent accidents on the road.

12.REFERENCES

[1] Drowsy Driving NHTSA reports. (2017, June 02). Retrieved from https://www.nhtsa.gov/risky-driving/drowsy-driving.

[2] K. Fagerberg, “Vehicle-based detection of inattentive driving for integration in an adaptive lane departure warning system Drowsiness detection,” M.S. thesis, KTH Signals Sensors and Systems, Stockholm, Sweden, 2004

[3] J. Krajewski, D. Sommer, U. Trutschel, D. Edwards and M. Golz, "Steering Wheel Behavior Based Estimation of Fatigue", in Proceedings of the Fifth International Driving Symposium on Human Factors in Driver Assessment, Training and Vehicle Design, pp. 118-124

[4] K. Mattsson, “In-vehicle prediction of truck driver sleepiness. Lane positions variables,” M.S. thesis, Division of Media Technology, Dept. of Computer Science and Electrical Engineering, Lulea Univ. of Technology, Sodertalje, Sweden, 2007.

[5] H. Malik, F. Naeem, Z. Zuberi, and R. ul Haq, “Vision based driving simulation,” in Proc. 2004 Int. Conf. Cyberworlds, 18–20 Nov. 2004, pp. 255–259

[6] Driver Alert Control (DAC). (2016, Feb 10). Retrieved from http://support.volvocars.com/uk/cars/Pages/owners-manual. aspx?mc=Y555&my=2015&sw=14w20&article=2e82f6fc0 d1139c2c0a801e800329d4e

[7] Z. Mardi, S. N. Ashtiani, and M. Mikaili, “EEG-based drowsiness detection for safe driving using chaotic features and statistical tests,” J. Med. Signals Sens., vol. 1, pp. 130–137, 2011.

[8] T. Danisman, I.M. Bilasco, C. Djeraba and N. Ihaddadene, “Drowsy driver detection system using eye blink patterns,” Universite Lille 1 & Telecom Lille 1, Marconi, France, 2010.

[9] B. Hariri, S. Abtahi, S. Shirmohammadi, and L. Martel, “A yawning measurement method to detect driver drowsiness,” Distrib. Collab. Virtual Environ. Res. Lab., Univ. Ottawa, Ottawa, ON, Canada, 2011

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. IEEE Conference on Computer Vision and Pattern Recognition, 2015.

[11] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In NIPS, 2015.

[12] B.-K. Kim, J. Roh, S.-Y. Dong, and S.-Y. Lee, Hierarchical committee of deep convolutional neural networks for robust facial expression recognition, Journal on Multimodal User Interfaces, 10 (2016), pp. 173–189.

[13] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 3431-3440.

[14] K. Dwivedi, K. Biswaranjan, A. Sethi, “Drowsy Driver Detection using Representation Learning,” Advance Computing Conference (IACC), 2014 IEEE International, 21-22 Feb. 2014.

[15] S. Park, F. Pan, S. Kang and C. D. Yoo, “Driver drowsiness detection system based on feature representation learning using various deep networks,” The ACCV Workshop on Driver Drowsiness Detection from Video 2016, Taipei, Taiwan, ROC, 2016.

[16] Krizhevsky, A., Sutskever, I., Hinton, G. E.: Imagenet classification with deep convolutional neural networks. NIPS, (2012) 1097–1105

[17] Parkhi, O. M., Vedaldi, A., Zisserman, A.: Deep face recognition. BMVC, 1 (2015) 6

[18] Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., Darrell, T.: Long-term recurrent convolutional networks for visual recognition and description. CVPR, (2015) 2625–2634

[19] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding," Computing Research Repository (CoRR), 2015.

[20] P. Gysel, M. Motamedi, and S. Ghiasi, "Hardware-Oriented Approximation of Convolutional Neural Networks," ICLR Workshop 2016.

[21] M.Courbariaux,Y.Bengio,Binarynet:Trainingdeepneural networks with weights and activations constrained to +1 or -1, CoRR abs/1602.02830, 2016

[22] Deep Networks on classification tasks using Torch. (2016, March 21). Retrieved from https://github.com/itayhubara/ BinaryNet

[23] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnornet: Imagenet classification using binary convolutional neural networks. In ECCV, pages 525–542. Springer, 2016.

[24] M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up convolutional neural networks with low rank expansions. In BMVC, 2014.
[25]G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network, " NIPS 2014, Dec. 2014.

[26] R. Caruana, A. N. Mizil, G. Crew, and A. Ksikes, “Ensemble Selection from Libraries of Models,” ICML ’04, 2004.
[27] J. Ba, and R. Caruana, “Do Deep Nets Really Need to be Deep,” NIPS 2014, Dec. 2014.
[28] B. B. Sau, and V. N. Balasubramanian, “Deep Model Compression: Distilling Knowledge from Noisy Teachers,” Computing Research Repository (CoRR) 2016.
[29] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, “FitNets: Hints for Thin Deep Nets,” Computing Research Repository (CoRR) 2014
[30] A. Romero, N. Ballas, S. Ebrahimi Kahou, A. Chassang, C. Gatta and Y. Bengio, “FitNets: Hints for Thin Deep Nets”, ICLR 2015.

[31] B. B. Sau and V. N. Balasubramanian, “Deep Model Compression: Distilling Knowledge from Noisy Teachers,” arxiv.org, 2016.

[32] K. Zhang, Z. Zhang, Z. Li, “Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks”, IEEE Signal Processing Letters, 2016

[33] K. Krafka, A. Khosla, P. Kellnhofer, H. Kannan, “Eye Tracking for Everyone”, CVPR 2016

[34] Q. Massoz, T. Langohr, C. Francois, J. G. Verly, “The ULg Multimodality Drowsiness Database (called DROZY) and Examples of Use, WACV 2016

[35] Ching-Hua Weng, Ying-Hsiu Lai, Shang-Hong Lai, “Driver Drowsiness Detection via a Hierarchical Temporal Deep Belief Network”, In Asian Conference on Computer Vision Workshop on Driver Drowsiness Detection from Video, Taipei, Taiwan, Nov. 2016

[36] Szegedy, Christian, et al. "Going deeper with convolutions." CVPR 2015.

[37] Liao, Shengcai, Anil K. Jain, and Stan Z. Li. "Partial face recognition: Alignment-free approach." IEEE Transactions on pattern analysis and machine intelligence 35.5 (2013)

[38]K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," ICLR 2015.