

A METHOD DRIVER'S EYES CLOSURE AND YAWNING DETECTION BY INFRARED CAMERA

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

SHANMUKH PRANEETH S (Reg.No - 39110901)
SUBHASH R (Reg.No – 39110819)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600119

APRIL - 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Shanmukh PraneethS (Reg.No - 391100901)** and **SUBHASH R (Reg.No – 39110819)** who carried out the Project Phase-2 entitled **“A METHOD DRIVER’S EYES CLOSURE AND YAWNING DETECTION BY INFRARED CAMERA”** under my supervision from January 2023 to April 2023.

Internal Guide

Dr. G. KALAIARASI, M.E., Ph.D.,

Head of the Department Dr.

L. LAKSHMANAN, M.E., Ph.D.,



Submitted for Viva voce Examination held on 20.04.2023

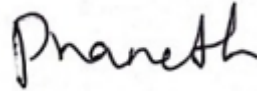
Internal Examiner

External Examiner

DECLARATION

I, **Shanmukh Praneeth S(Reg.No - 391100901)**, hereby declare that the Project Phase-2 Report entitled **“A METHOD DRIVER’S EYES CLOSURE AND YAWNING DETECTION BY INFRARED CAMERA”** done by me under the guidance of **Dr. Dr.G.KALAIARASI, M.E.,Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE: 20.04.2023

A handwritten signature in black ink, appearing to read 'Praneeth', is written over a faint, circular official stamp.

PLACE: Chennai

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D**, Dean, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.G.KALAIARASI M.E.,Ph.D.**,for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase-2 project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Diversions can result in hazardous incidents that are challenging to prevent as the world's population rises. Many studies have been conducted in an effort to figure out how to stop the death toll. The designers of the car put the safety of the occupants first. Airbags are made to increase security and save lives of occupants, but they do not prevent accidents from happening. The primary causes are general exhaustion and distraction from phone alerts. In the past, a wide range of methods based on behavioral measurements and machine learning approaches have been examined to identify driver distraction. These algorithms are necessary given the rapid expansion of such technology. The greatest frequent danger to their lives and the lives of others nowadays is driving while intoxicated. Although we cannot help someone stop being sleepy, we can make them awake and alert while they are driving. The suggested model was created by a few persons utilizing various technologies. But in order to increase the detection's precision, this model was created using specific technologies, namely OpenCV and Keras. A CNN model developed using Keras will be trained using the dataset after the driver's face is detected using OpenCV and a Haar Cascade Classifier. This trained model will output whether the driver is alert or tired based on the input of the detected face. Convolution neural networks are used in this study's new methodology to identify the distraction and immediately play a loud siren to provide a sensory shock that restores attentiveness. The aforementioned hybrid strategy would result in the greatest immediate resolution to such problems in the future when combined with appropriate instruction. A Convolutional Neural Network capable of identifying the eye and mouth detection in a given image is built, trained, and tested using Keras and TensorFlow of which 2000 are drowsiness images and 2000 are yawning images, make up this dataset. These photos are loaded, then transformed into a NumPy array and normalized to have the best precision.

TABLE OF CONTENTS

S. NO.		Pg NO.
	ABSTRACT	v
	LIST OF FIGURES	viii
	LIST OF TABLES	ix
	LIST OF ABBREVIATIONS	x
1	INTRODUCTION	1
2	LITERATURE SURVEY	5
	2.1 Inferences from Literature Survey	8
	2.2 Open problems in Existing System	10
3	REQUIREMENTS ANALYSIS	12
	3.1 Feasibility study/Risk analysis of the project	12
	3.2 Software Requirements Specification Document	17
	3.2.1 Functional Requirements	19
	3.2.2 Non-Functional Requirements	19
	3.2.3 Environmental Requirements	19
	3.3 Methods and Algorithms Used	20
	3.3.1 Python Packages	20
	3.3.2 Keras and Tensor Flow	22
	3.3.3 CNN Architecture	23
	3.3.4 Parameters in CNN Model	29
4	DESCRIPTION OF PROPOSED SYSTEM	31
	4.1 Selected Methodology or Process Model	31
	4.2 Architecture / Overall Design of Proposed System	32
	Description of Software For Implementation and Testing	33
	4.3 Plan of the Proposed Model/System	33
	4.4 Project Management Plan	35
	4.5 Financial Report On Estimated Costing	36
	4.6 Transition/ Software to Operations Plan	37

5	IMPLEMENTATION DETAILS	39
	5.1 Development and Deployment Setup	39
	5.1.1 Dataset Description	40
	5.1.2 Data Preprocessing	41
	5.2 Algorithms	42
	5.3 Library	45
6	RESULTS AND DISCUSSION	47
7	CONCLUSION	52
	7.1 Conclusion	52
	7.2 Future Work	52
	7.3 Research Issues	53
	7.4 Implementation Issues	54
	REFERENCES	56
	APPENDIX	58
	A. SOURCE CODE	58
	B. SCREENSHOTS	63
	C. RESEARCH PAPER	67

LIST OF FIGURES

FIGURE NO	FIGURE NAME	Page No.
1.1	STATISTICS OF ROAD ACCIDENT	1
1.2	EXAMPLE OF FATIGUE & DROWSINESS CONDITION	2
3.1	CONVOLUTIONAL NEURAL NETWORK	22
3.2	CONVOLUTIONAL NETWORK	23
3.3	MAX POLING	24
3.4	AVERAGE POLING	24
3.5	GRAPH OF RELU ACTIVATION FUNCTION	25
3.6	GRAPH OF SOFTMAX ACTIVATION FUNCTION	25
3.7	GRAPH OF THE ADAM OPTIMIZER	26
3.8	DROPOUT LAYER	27
4.1	SYSTEM ARCHITECTURE	30
5.1	DROWSINESS DATA SET	38
5.2	YAWNING DATA SET	38
5.3	CONVOLUTIONAL NEURAL NETWORK	41
5.4	HAAR CASCADE ALGORITHM	42
5.5	OPEN CV	42
6.1	ACCURACY OF EPOCHS	45
6.2	TESTING OF DATA SET	45
6.3	GRAPH OF ACCURACY	46
6.4	GRAPH OF LOSS	46
6.5	OUTPUT OF ACTIVE STATUS	46
6.6	OUTPUT OF YAWN ALERT	47
6.7	OUTPUT OF DROWSY STATUS	47
6.8	NORMAL DRIVER OUTPUT	47
6.9	DROWSY DRIVER OUTPUT	48

LIST OF TABLES

SR NO.	NAME OF TABLE	PAGE NO
1	ACCURACY TABLE	53

LIST OF ABBREVIATIONS

ABBREVIATIONS	EXPANSION
ML	Machine Learning
CNN	Convolution Neural Network
CV	Computer Vision
RELU	Rectified Linear Unit
SGD	Stochastic Gradient Descent
ROI	Region of Interest
API	Application Programing Interface
HOG	Histogram of Oriented Gradients
PERCLOS	Percentage of Eye Closure
MIROS	Malaysia Institute of Road Safety
EEG	Electroencephalography

CHAPTER 1

INTRODUCTION

Drowsiness is a state of near sleep, where the person has a strong desire for sleep. It has two distinct meanings, referring both to the usual state preceding falling asleep and the chronic condition referring to being in that state independent of a daily rhythm. Sleepiness can be dangerous when performing tasks that require constant concentration, such as driving a vehicle. When a person is sufficiently fatigue while driving, they will experience drowsiness, and this leads to increase the factor of road accident.

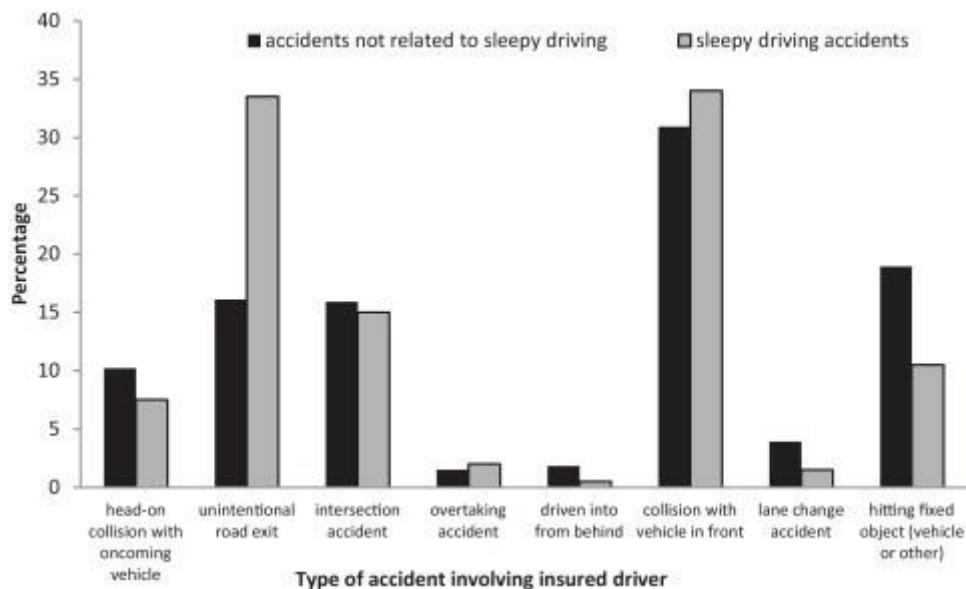


Fig 1.1: STATISTIC OF ROAD ACCIDENT

Figure 1 shows the statistic of road accident in Malaysia from the year 2005 to 2023 provided by MIROS (Malaysia Institute of Road Safety). The numbers of vehicles involved in road accident keep increasing each year. From Figure 1, car and taxi type of vehicles shows about nearly 400,000 cases of road accident has been recorded. It keeps increasing every year and by the year 2009, it shows the number of road accident were recorded by MIROS are nearly 500,000. These are significant and latent dangers responsible for much loss of lives. In recent years, scientists have been trying to prevent any further loss by pre-emptively spotting such symptoms well in advance. These recognizing methods are characterized as

subjective and objective detection. In the subjective detection method, a driver must participate in the evaluation, which is associated with the driver's subjective perceptions through steps such as self- questioning.

Figure 1.2 shows the difference between fatigue and drowsiness condition.



Fig 1.2: EXAMPLE OF FATIGUE & DROWSINESS CONDITION

The development of technologies for detecting or preventing drowsiness while driving is a major challenge in the field of accident avoidance system. Because of the hazard that drowsiness presents on the road, methods need to be developed for counteracting its affects.

The aim of this project is to develop a simulation of drowsiness detection system. The focus will be placed on designing a system that will accurately monitor the open or closed state of the driver's eyes and mouth. By monitoring the eyes, it is believed that the symptoms of driver's drowsiness can be detected in sufficiently early stage, to avoid a car accident. Yawning detection is a method to assess the driver's fatigue. When a person is fatigue, they keep yawning to ensure that there is enough oxygen for the brain consumption before going to drowsiness state . Detection of fatigue and drowsiness involves a sequence of images of a face, and the observation of eyes and mouth open or closed duration. Another method to detect eye closure is PERCLOS. This detection method is based on the time of eyes closed which refers to percentage of a specific time.

The analysis of face images is a popular research area with applications such as face recognition, and human identification and tracking for security systems. This project is focused on the localization of the eyes and mouth, which involves looking at the entire image of the face, and determining the position of the eyes and mouth, by applying the existing methods in image processing algorithm. Once the position of the eyes is located, the system is designed to determine whether the eyes and

mouth are opened or closed, and detect fatigue and drowsiness.

Each year, there is an increase in road accidents cases involving cars and heavy vehicles like buses, lorries and trucks in Malaysia. Drowsiness and fatigue condition is one of the prime factors contributing to road accidents. Driving in this condition may result terrible causes since it affects the driver's judgment and concentration. Falling asleep on the wheel can be avoid if the drivers take efforts such as getting enough sleep before driving, taking caffeine or stop for a while to rest when the signs of fatigue and drowsiness appears.

In recent years, scientists have been trying to prevent any further loss by pre-emptively spotting such symptoms well in advance. These recognizing methods are characterized as subjective and objective detection. In the subjective detection method, a driver must participate in the evaluation, which is associated with the driver's subjective perceptions through steps such as self- questioning. Then, these data are used to estimate the danger of the vehicles being driven by exhausted drivers, assisting them to plan their schedules accordingly. However, their feedback is not required in the objective detection method as it monitors their physiological state and driving-behavior characteristics in real time. The collected data are used to evaluate the driver's level of fatigue. Furthermore, objective detection is categorized into two: contact and non-contact. Compared with the contact method, noncontact is cheaper and more convenient because it only requires Computer Vision technology with sophisticated camera which allows the use of the device in large numbers. Owing to easy installation and low cost, the noncontact method has been widely used for our problem statement. For instance, Attention Technologies and Smart Eye observe the movement of the driver's eyes and position of the driver's head to determine the level of their fatigue. In this study, we propose a non-contact method to detect the level of the driver's fatigue. Our method employs the use of only the vehicle- mounted camera, making it unnecessary for the driver to carry any on/in-body devices. Our design uses each frame image to analyze and detect the driver's concentration state.

However, in many cases, drivers refuse to take one of these steps even when they know that they are suffering from fatigue, and will continue driving. Therefore, detecting drowsiness is important as one of the steps to prevent the road accidents.

This project proposed that yawning and eyes detection is the obvious signs of fatigue and drowsiness. when the driver is getting lazy. Different considerations have been proposed that around 20% of all street mishaps are fatigue-related, up to 50% on certain streets. Driver weakness could be a critical reason in the large number of vehicle mishaps. Later measurements assess that yearly 1,200 passing's and 76,000 wounds can be credited to weariness related crashes. Vehicle users in our country are growing exponentially. The biggest problem with increasing vehicle use is the increase in road accidents. Global Status Report on road safety published by the World Health Organization (WHO) identified the major cause of road accidents is due to drowsiness of the driver. Developing technologies to detect or prevent drowsiness is a major challenge in the field of accident avoidance systems .Now a days "TATA TRUCKS" also started using these technologies to prevent accidents. Here we built our model with CNN and OpenCV.

CHAPTER 2

LITERATURE SURVEY

This problem statement has been extensively studied over the past 5 years by researchers and automotive companies in a bid to create a solution, and all their solutions vary from analyzing various patterns of distractive habits to analyzing health vitals of the driver.

The paper titled "IoT based driver drowsiness detection and health Monitoring System" by Tiwari et al[1]. was published in the International Journal of Research and Analytical Reviews in 2019. The paper proposes a system that uses IoT-based technology to monitor the driver's health and detect drowsiness while driving. The system consists of multiple sensors, including a heart rate sensor, a temperature sensor, a humidity sensor, and a camera-based drowsiness detection module. The authors tested the system in a simulated driving environment and found that it was effective in detecting drowsiness and monitoring the driver's health. The system was also found to be accurate and reliable in detecting drowsiness, with a detection accuracy of 97%.

The paper titled "Driver Drowsiness Detection Model Using Convolutional Neural Networks Techniques for Android Application" by Jabbar et al[2]. was published in the proceedings of the 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT). The paper proposes a driver drowsiness detection model based on convolutional neural networks (CNN) for an Android application. The proposed model uses a dataset of images of drivers to train a CNN model to detect driver drowsiness based on features such as eye closure and head movement. The authors evaluated the performance of the proposed model using an Android application that captures real-time images of the driver and applies the trained CNN model to detect driver drowsiness. The evaluation results showed that the proposed model achieved an accuracy of 92.5% in detecting driver drowsiness. The paper also discusses the potential benefits of the proposed model in improving road safety and preventing accidents caused by driver drowsiness. The authors suggest that the proposed model can be integrated into existing driver

assistance systems to provide real-time alerts to drivers who show signs of drowsiness.

The paper titled "IoT based real-time drowsy driving detection system for the prevention of road accidents" by Hossain and George[3] was presented at the 2018 International Conference on Intelligent Informatics and Biomedical Sciences. The paper proposes a real-time drowsy driving detection system that utilizes the Internet of Things (IoT) technology to prevent road accidents. The system consists of a drowsiness detection module that uses a camera to monitor the driver's eyes and facial features for signs of drowsiness, and an alert module that triggers an alarm to alert the driver when drowsiness is detected. The authors tested the system in a simulated driving environment and achieved a detection accuracy of 95%. The system was also found to be effective in preventing accidents caused by drowsy driving.

The paper titled "Comprehensive drowsiness level detection model combining multimodal information" by Sunagawa et al[4]. was published in the IEEE Sensors Journal in 2019. The paper proposes a comprehensive drowsiness level detection model that combines multiple modalities of information, including physiological signals, eye movement, and driving behavior. The model uses machine learning algorithms to analyze the data and detect the driver's level of drowsiness accurately. The authors conducted experiments with the proposed model and found that it outperformed existing models in terms of accuracy and robustness. The proposed model was also able to detect drowsiness levels accurately even in situations where the driver's eyes were not visible.

The paper titled "Detecting human driver inattentive and aggressive driving behavior using deep learning: Recent advances, requirements and open challenges" by Alkinani et al[5]. was published in IEEE Access in 2020. The paper provides an overview of recent advances in detecting human driver inattentive and aggressive driving behavior using deep learning. The authors discuss the various types of data that can be used for this purpose, including video, audio, and physiological signals. The paper also discusses the requirements for developing accurate and reliable detection systems, including the need for large datasets, appropriate feature

extraction techniques, and effective deep learning models. The authors also highlight the open challenges in this field, including the need for more diverse datasets, better feature extraction techniques, and more robust deep learning models.

The paper titled "Driver drowsiness detection using behavioral measures and machine learning techniques: A review of state-of-art techniques" by Ngxande et al[6]. was presented at the 2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech) conference. The paper provides a review of the state-of-the-art techniques for driver drowsiness detection using behavioral measures and machine learning techniques. The authors discuss various methods for measuring driver behavior, including eye tracking, facial expression analysis, and physiological signals such as heart rate variability. The paper also discusses the various machine learning techniques used for driver drowsiness detection, including neural networks, decision trees, and support vector machines. The authors highlight the advantages and limitations of each technique and provide recommendations for future research in this area.

The paper titled "Cross-Subject Zero Calibration Driver's Drowsiness Detection: Exploring Spatiotemporal Image Encoding of EEG Signals for Convolutional Neural Network Classification" by Paulo et al[7]. was published in the IEEE Transactions on Neural Systems and Rehabilitation Engineering in 2021. The paper proposes a novel method for detecting driver drowsiness using electroencephalography (EEG) signals and a convolutional neural network (CNN). The authors develop a cross- subject zero calibration approach to improve the robustness of the CNN classification model and explore spatiotemporal image encoding of EEG signals for feature extraction. The authors conducted experiments with the proposed method and compared its performance with other state-of-the-art methods. The results showed that the proposed method outperformed existing methods in terms of accuracy and robustness.

The paper titled "Multimodal System to Detect Driver Fatigue Using EEG, Gyroscope, and Image Processing" by Karuppusamy and Kang[8] was published in

IEEE Access in 2020. The paper proposes a multimodal system for detecting driver fatigue using EEG, gyroscope, and image processing. The system uses a wireless EEG sensor to capture brainwave signals, a gyroscope sensor to measure head movements, and a camera to capture facial images for analysis. The authors develop a machine learning algorithm that integrates the data from these sensors and applies image processing techniques to detect signs of fatigue in the driver. The proposed algorithm achieved an accuracy of 96% in detecting driver fatigue. The paper also presents a dataset that was collected to validate the proposed system. The dataset consists of EEG, gyroscope, and image data collected from 16 drivers under different driving conditions.

The paper titled "Driver Drowsiness Recognition via 3D Conditional GAN and Two-Level Attention Bi-LSTM" by Hu et al[9]. was published in IEEE Transactions on Circuits and Systems for Video Technology in 2019. The paper proposes a novel approach to detecting driver drowsiness using a 3D conditional generative adversarial network (GAN) and a two-level attention bidirectional long short-term memory (Bi-LSTM) network. The proposed method captures the spatiotemporal information from multiple modalities, including images and physiological signals, to recognize driver drowsiness. The authors conducted experiments with the proposed method using a dataset consisting of both simulated and real-world driving scenarios. The results show that the proposed method achieved an accuracy of 93.26% in detecting driver drowsiness, outperforming several state-of-the-art methods. The paper also discusses the importance of attention mechanisms in detecting driver drowsiness. The proposed two-level attention Bi-LSTM network focuses on important features and learns to ignore irrelevant parts of the input data, improving the accuracy of the detection.

The paper titled "Internet of Things Based Intelligent Drowsiness Alert System" by Bala and Sarath[10] was published in the proceedings of the 2020 5th International Conference on Communication and Electronics Systems (ICCES). The paper proposes an Internet of Things (IoT) based intelligent drowsiness alert system that utilizes sensors to detect the driver's drowsiness and alert the driver before an accident occurs. The proposed system consists of a microcontroller, a drowsiness detection unit, a GSM module, and an alert unit. The drowsiness detection unit in the

proposed system uses a sensor to detect the driver's eyelid movements and a microphone to detect the driver's snoring sounds. If the driver shows signs of drowsiness, the alert unit in the system emits an alarm and sends an alert message to a preconfigured phone number using the GSM module. The authors conducted experiments with the proposed system and achieved a detection accuracy of 95% in detecting driver drowsiness. They also discussed the potential benefits of the proposed system in improving road safety and preventing accidents caused by driver drowsiness.

2.1 INFERENCES FROM LITREATURE SURVEY

- [1] The paper highlights the potential of IoT-based technologies in improving road safety and preventing accidents caused by drowsy driving. The system proposed in the paper provides a comprehensive approach to driver monitoring and can help prevent accidents caused by drowsiness and other health-related issues.
- [2] The paper presents a practical solution for detecting driver drowsiness using CNN-based technology for an Android application. The proposed model has the potential to improve road safety and prevent accidents caused by driver drowsiness.
- [3] The pape0.2 highlights the potential of IoT-based technologies in improving road safety and reducing the number of accidents caused by drowsy driving.
- [4] The paper highlights the potential of combining multiple modalities of information for detecting drowsiness levels accurately. The proposed model provides a comprehensive approach to drowsiness detection, which can help prevent accidents caused by drowsy driving.
- [5] The paper provides a comprehensive overview of the state-of-the-art in detecting human driver inattentive and aggressive driving behavior using deep learning. The authors highlight the potential of this approach to improve road safety and reduce the number of accidents caused by inattentive or aggressive driving.
- [6] The paper provides a comprehensive overview of the state-of-the-art in driver drowsiness detection using behavioral measures and machine learning techniques. The authors highlight the potential of this approach to improve road

safety and reduce the number of accidents caused by drowsy driving.

- [7] The paper highlights the potential of using EEG signals and CNNs for driver drowsiness detection. The proposed method provides a promising approach to improving road safety by detecting driver drowsiness accurately and reliably.
- [8] The paper provides a promising approach to detecting driver fatigue using a multimodal system. The proposed system can potentially improve road safety by alerting drivers when they are becoming fatigued and at risk of causing accidents.
- [9] The paper provides a promising approach to detecting driver drowsiness using a multimodal system that incorporates attention mechanisms. The proposed method can potentially improve road safety by alerting drivers when they are becoming drowsy and at risk of causing accidents.
- [10] The paper presents a practical solution for detecting driver drowsiness using IoT-based technology. The proposed system has the potential to improve road safety and prevent accidents caused by driver drowsiness.

2.2 OPEN PROBLEMS IN EXISTING SYSTEM

Wisaroot and Charoenpong produced the paper Identifying the driver's face and designating a region of interest for it are the important concepts (ROI). Next, utilize ROI to determine mouth and eye targets. This process starts with data collection from an infrared 2D camera and processing in MATLAB R2015a.

The five steps in the flowchart are image acquisition, face detection, eye detection, mouth detection, and eyes closed and yawning detection. One technique that can recognize faces is the Haar-like feature. Michael Jones and Paul Viola are the authors of this technique. In this stage, an area of interest (ROI) is created in order to locate the driver's face. The dominant facial feature is classified using the Harr-like feature, which compares threshold and polarity to the difference between the shading rectangle and the regular rectangle

The author had to divide the face image into two half. This technique limits the ability to see the driver's eyes and mouth. The camera settings cause the driver's picture to be slightly skewed, thus this stage rotates the image by around three degrees to the left and right in order to detect the driver's eyes and mouth using the Haar like characteristic.

Author usage of classify data machine that supports vectors (SVM). Normal photos must be converted into vector data by obtaining the Histogram of orientated images prior to training the SVM.

(HOG). HOG is a technique that divides a picture into several cells, gathers histograms from each gradient, and calculates the scale and vector of each cell . The HOG of each pixel is a vector.

The following stage includes two groups of data, such as eyes open and closed. SVM comes in a variety of forms. SVM is appropriate for this paper. SVM's guiding principles are learning models for data analysis.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDIES/RISK ANALYSIS OF THE PROJECT

The project plan and objective of the project is clear and this project is implemented till application level without any problem. Also, the risk involved in this project is very minimal which is negligible. This project can further be carried out in next phase as well it has wide scope. As currently four classification algorithms are used in project. The accuracy and performance of each algorithm is used further to predict with the client-side data and it can also be improved in future. The applications is built with minimal risk. So, from this we can analyze that the project and feasible and it is submitted on time. The theme of the feasibility study is to find out the application is technical, economical, operational and market feasible so that applications sustain for longer time and gives return on investment with good results. It can be used to inform future decisions.

Feasibility Studies:

- **Technical Feasibility:** The project's technical feasibility depends on the availability of necessary hardware, software, and data infrastructure. It is important to evaluate whether the required technology and data is readily available and can be obtained within a reasonable time and cost.
 - i. **Eye-tracking technology:** Eye-tracking technology can be used to detect drowsiness by tracking the movements of a person's eyes. As a person becomes drowsy, their eye movements may become slower and less frequent. Eye-tracking technology can detect these changes and alert the person or an external system.
 - ii. **Wearable devices:** Wearable devices such as smartwatches or fitness trackers can be used to monitor a person's heart rate, breathing rate, and other physiological indicators. Changes in these indicators can indicate drowsiness and trigger an alert.

- iii. Machine learning algorithms: Machine learning algorithms can be trained to detect patterns in a person's behavior that are indicative of drowsiness. For example, an algorithm might be able to detect when a person's typing speed slows down or when they start making more errors.
- iv. Video analysis: Video analysis techniques can be used to detect drowsiness by analyzing a person's facial expressions and body movements. For example, a person who is becoming drowsy may start to slouch in their chair or their head may droop.
- v. Brain wave analysis: Brain wave analysis techniques, such as electroencephalography (EEG), can be used to detect changes in a person's brain activity that indicate drowsiness.

The project's technical feasibility depends on the availability of necessary hardware, software, and data infrastructure. It is important to evaluate whether the required technology and data is readily available and can be obtained within a reasonable time and cost.

- Operational Feasibility: In the context of a drowsiness detection , operational feasibility refers to the practicality and effectiveness of implementing the project in a real-world setting.

Here are some factors that can affect the operational feasibility of a drowsiness detection project:

- i. Technical feasibility: The project should be technically feasible, meaning that the technology used to detect drowsiness should be accurate and reliable. It is important to ensure that the detection system can differentiate between actual drowsiness and other factors that may affect a person's behavior, such as distractions or stress.
- ii. Cost feasibility: The project should also be financially feasible, taking into account the cost of developing and implementing the drowsiness detection system. The cost should be reasonable and justifiable given the benefits of the project.
- iii. User acceptance: The project should be acceptable to users, including drivers or operators who will be using the drowsiness detection system. The

system should be easy to use and should not cause any inconvenience or discomfort to the user.

- iv. Legal and regulatory compliance: The project should comply with relevant laws and regulations related to driver safety, privacy, and data protection.
- v. Maintenance and support: The project should be easy to maintain and support, and there should be a plan in place to address any issues that arise during the operation of the system.

In summary, operational feasibility is a critical consideration when developing a drowsiness detection project. The project should be technically feasible, financially feasible, acceptable to users, compliant with legal and regulatory requirements, and easy to maintain and support. By addressing these factors, the project can be successfully implemented in a real-world setting to improve driver safety.

Risk Analysis:

- Technical Risks: There are several technical risks associated with a drowsiness detection project. Here are some of them:
 - i. Accuracy of the detection system: One of the primary technical risks is the accuracy of the drowsiness detection system. The system needs to be able to accurately detect drowsiness in real-time to prevent accidents. If the system is not accurate, it may lead to false alarms or worse, not detecting drowsiness when it is present.
 - ii. Robustness of the detection system: Another technical risk is the robustness of the system. The detection system should be able to work effectively under various conditions, such as different lighting conditions, head positions, and other environmental factors. If the system is not robust enough, it may fail to detect drowsiness in some situations.
 - iii. Integration with other systems: The drowsiness detection system may need to be integrated with other systems, such as a vehicle's alert system or a wearable device. The integration process can be complex and may require a significant amount of testing to ensure that the systems work together effectively.

- iv. Processing time: The system needs to process data in real-time to detect drowsiness effectively. If the processing time is too long, the system may not be able to detect drowsiness quickly enough to prevent accidents.
 - v. Power consumption: The detection system may need to be powered by a battery, so power consumption is an essential factor to consider. If the system consumes too much power, it may drain the battery quickly, making it less practical for long-term use.
 - vi. Scalability: If the system is designed for use in a single vehicle, it may be challenging to scale it up for use in a fleet of vehicles or other applications. The scalability of the system needs to be considered during the design and development process.
 - vii. False alarms: Finally, the drowsiness detection system should avoid generating false alarms as they may lead to driver frustration and annoyance, leading to the potential for human error.
- Data Risks: The use of data in drowsiness detection can introduce several risks, including:
 - i. Privacy risks: Drowsiness detection systems often collect sensitive information such as facial expressions, eye movements, and biometric data. This information can be exploited if it falls into the wrong hands, potentially leading to identity theft, discrimination, or other forms of harm.
 - ii. Accuracy risks: Drowsiness detection algorithms rely on accurate and representative data to function properly. If the data used to train the algorithm is biased or incomplete, the resulting system may produce inaccurate or unfair results, leading to false alarms or missed detections.
 - iii. Security risks: Drowsiness detection systems that are connected to the internet or other networks may be vulnerable to cyberattacks, which could compromise the integrity or confidentiality of the data they collect or transmit.
 - iv. Reliability risks: Drowsiness detection systems may produce false positives or false negatives, leading to unnecessary alerts or missed incidents. This can be particularly dangerous in safety-critical applications, such as driving or operating heavy machinery.

- v. Ethical risks: The use of drowsiness detection systems raises ethical questions about privacy, consent, and autonomy. It is important to consider the potential impact of these systems on individuals and society as a whole, and to ensure that they are developed and deployed in a responsible and transparent manner.
- Adoption Risks: Adoption risks for a drowsiness detection can be both technical and non-technical. Here are some potential adoption risks to consider:
 - i. Accuracy: The accuracy of the drowsiness detection algorithm is critical to its adoption. If the algorithm does not accurately detect drowsiness, it can lead to false alarms or missed detections, which can reduce trust in the system and discourage its use.
 - ii. Compatibility: The drowsiness detection system needs to be compatible with different types of vehicles, as well as different operating systems and hardware configurations. If the system is not compatible with a wide range of devices, it can limit its adoption.
 - iii. Cost: The cost of implementing a drowsiness detection system can be a significant barrier to adoption. If the cost of the system is too high, it may not be feasible for small businesses or individual drivers to afford it.
 - iv. Privacy concerns: The collection and analysis of driver data can raise privacy concerns. If drivers feel that their privacy is being invaded, they may be hesitant to adopt the system.
 - v. Regulatory requirements: The implementation of a drowsiness detection system may be subject to regulatory requirements that can increase the complexity and cost of the system. Failure to comply with these requirements can result in legal consequences and reduced adoption.
 - vi. User experience: The user experience of the system is crucial to its adoption. If the system is difficult to use or requires significant effort from the driver, it can discourage adoption.
 - vii. Maintenance and support: The ongoing maintenance and support of the system can be a significant adoption risk. If the system is difficult to maintain or support, it can lead to increased downtime and decreased trust in the system.

- **Operational Risks:** Operational risks are those risks associated with the day-to-day operations of a project or business. In the case of a drowsiness detection , some of the operational risks that could arise include:
 - i. **False alarms:** The drowsiness detection system may generate false alarms, which could lead to driver annoyance or loss of confidence in the system.
 - ii. **Malfunctioning hardware:** The hardware components of the system, such as cameras, sensors, and microphones, may fail to function properly, leading to inaccurate detection of drowsiness or no detection at all.
 - iii. **Data privacy and security:** The drowsiness detection system may collect sensitive data about drivers, such as their facial expressions and eye movements, which could be at risk of being hacked or accessed by unauthorized persons.
 - iv. **Integration issues:** The system may not integrate well with existing vehicle systems, such as warning systems or automated driving systems, leading to reduced functionality or compatibility issues.
 - v. **Cost overruns:** The project may incur cost overruns due to unexpected hardware or software failures, additional development time, or unforeseen technical difficulties.
 - vi. **Maintenance and support:** The system may require ongoing maintenance and support, which could be costly and time-consuming.
 - vii. **User acceptance:** Drivers may not accept the system, either due to concerns about privacy or distrust of the technology, leading to low adoption rates and reduced effectiveness.

3.2 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

It is crucial to comprehend and record all forms of requirements to guarantee that the software system or application is created in a way that meets the expectations of the users and stakeholders and operates as anticipated within its designated environment. A project typically encompasses three types of requirements, namely functional requirements, non-functional requirements, and environmental requirements.

Purpose of this document:

The purpose of this document is to identify the essential requirements of the software and define the criteria to make sure the development is moving in the right path

Scope of this document:

The scope of drowsiness detection using CNN in transportation is quite significant as it has the potential to improve the safety of drivers and passengers on the road. Drowsy driving is a major cause of road accidents, and by detecting signs of drowsiness in drivers, accidents can be prevented.

Convolutional Neural Networks (CNNs) have shown great success in various computer vision tasks, including face detection and recognition. When applied to drowsiness detection, CNNs can analyze the driver's facial features and detect patterns that indicate drowsiness, such as drooping eyelids and a decrease in eye movements.

Drowsiness detection using CNNs can be applied to various modes of transportation, including cars, buses, and trains. It can also be used in different contexts, such as monitoring the drowsiness of commercial drivers or detecting fatigue in passengers who may be in charge of driving.

Overall, the scope of drowsiness detection using CNNs in transportation is promising, as it has the potential to significantly reduce the number of accidents caused by drowsy driving and improve road safety for everyone.

Requirements are the primary constraints that are required to develop a system. Requirements are collected while designing the system. The following are the requirements that are to be discussed.

1. Functional requirements
2. Non-Functional requirements
3. Environmental requirements

3.2.1 FUNCTIONAL REQUIREMENTS:

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis process. It lists the requirements of a particular software system.

3.2.2 NON-FUNCTIONAL REQUIREMENTS:

Process of functional steps:

1. Problem define
2. Preparing data
3. Evaluating algorithms
4. Improving results
5. Prediction of the result
6. Integration of model with Flask
7. Deployment using Flask

3.2.3 ENVIRONMENTAL REQUIREMENTS:

1. Software Requirements:

Operating system: Windows

Tool: Anaconda with Jupyter notebook, spyder

Updated Python libraries

2. Hardware Requirements:

- A. Internet connection to download and activate.
- B. Minimum 10GB free disk space
- C. Windows 8.1 or 10 (64-bit version only) is required.
- D. Minimum System Requirements To run Office Excel 2013, your computer needs to meet the following minimum hardware requirements:
 - 500-megahertz (MHz)
 - 256 megabytes (MB) RAM
 - 1.5 gigabytes (GB) available space
 - 1024x768 or higher resolution monitor

It describes the environment in which the software system or application will operate, including the hardware, software, and network infrastructure. They typically specify the resources required to deploy and run the system, such as hardware specifications, operating systems, and software dependencies. Examples of environmental requirements include the operating system, database management system, web server, network protocols, and required software libraries. An operating system is windows7+ or macOS 10.6+, anaconda and vs-code are the software requirement of this project to run it uninterruptable. As far as hardware is concerned a good SSD is recommended to run it smoothly but HDD is also fine to run the program. Minimum of 4GB RAM and 128GB storage is required.

3.3 METHODS AND ALGORITHMS USED:

Since this is a multi-class classification problem under deep learning with image recognition. We are going to use the CNN algorithm with Keras and TensorFlow methods. As classification is a supervised learning approach in which the computer program learns from the data input and images and uses these learnings to classify new observations. Some examples of classification problems are speech recognition, handwriting recognition, biometric identification, document classification, etc. In Supervised Learning, algorithms learn from labeled data. After understanding the data, the algorithm determines which label should be given to new data based on the pattern and associates the patterns with the unlabeled new data.

3.3.1 PYTHON PACKAGES:

- **NumPy:**

It is an open-source numerical Python library. It contains a multidimensional array and matrix data structures and can be used to perform mathematical operations.

- **Matplotlib and Seaborn:**

Matplotlib is mainly deployed for basic plotting. Visualization using Matplotlib generally consists of bars, pies, lines, scatter plots, and so

on. Seaborn: Seaborn, on the other hand, provides a variety of visualization patterns. It uses fewer syntax and has easily interesting default themes.

- **Pandas:**

It is a fast, powerful, flexible, and easy-to-use open-source data analysis and manipulation tool, built on top of the Python programming language. It is a popular open-source library for data manipulation and analysis in Python. It provides tools for working with structured data, such as tabular data and time series data.

- **Flask:**

Flask provides configuration and conventions, with sensible defaults, to get started. This section of the documentation explains the different parts of the Flask framework and how they can be used, customized, and extended. Beyond Flask itself, look for community-maintained extensions to add even more functionality.

- **Open-CV:**

OpenCV (Open-Source Computer Vision) is a computer vision library that provides tools for image and video processing, including image pre-processing and feature extraction. It is widely used for object recognition, face detection, and motion tracking, among other applications.

- **PyTorch:**

Torch vision is a PyTorch package that provides a range of tools and utilities for working with computer vision tasks, including image classification, object detection, and segmentation. It includes a number of pre-trained models, as well as tools for loading and pre-processing image data. These models have been trained on large datasets, such as ImageNet, and can be fine-tuned on new datasets with relatively few additional training examples.

- **Scikit-learn:**

It is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction in Python.

3.3.2 KERAS AND TENSORFLOW

- **Keras:**

Keras is an open-source deep learning framework written in Python. It provides a user-friendly interface for building and training neural networks. Keras was designed to be user-friendly, modular, and extensible, which makes it a popular choice among researchers and developers.

Keras is a high-level python API that can be used to quickly build and train neural networks using either TensorFlow or Theano as the back end. Keras is one of the fastest-growing libraries for deep learning. The main data structure in Keras is the model which provides a way to define the complete graph. You can add layers to the existing model/graph to build the network you want.

Keras has two distinct ways of building models:

- I. **Sequential models:** This is used to implement simple models. You simply keep adding layers to the existing model.
- II. **Functional API:** Keras functional API is very powerful and you can build more complex models using it, models with multiple outputs, directed acyclic graphs, etc.

- **TensorFlow:**

TensorFlow operates by representing computational graphs as a series of mathematical operations that can be executed on multiple devices, including CPUs and GPUs. This allows developers to scale their models to large datasets and distributed systems. TensorFlow also provides a wide range of tools and libraries to support machine learning development, including pre-built models for image and speech recognition, natural language processing, and reinforcement learning. Additionally, TensorFlow can be used in a variety of programming languages, including Python, C++, and Java.

It is designed to enable developers to create and deploy large-scale machine learning models for a variety of applications. TensorFlow provides a flexible architecture that allows developers to create and customize machine learning models, from simple linear regression to complex neural networks. TensorFlow

allows developers to create dataflow graphs and structure that describes how data moves through a graph (or) a series. Overall, TensorFlow is a powerful and flexible framework that enables developers to create and deploy machine learning models quickly and efficiently. Its versatility and scalability make it an excellent choice for both research and production-level applications.

3.3.3 CNN ARCHITECTURE USING KERAS AND TENSORFLOW

A convolutional neural network or CNN is a fascinating topic and it has many real-world and highly useful applications, such as self-driving cars, facial recognition, object detection, and motion or movement detection, etc., Neural networks are the main building blocks of deep learning and the convolutional neural networks are situated on top of it. The main idea behind the convolutional neural network is using filters. The filters are responsible for detecting objects or images. These filters are also useful to detect the edges around the image. These edges will be passed to the next layer to detect the ears, eyes, nose, etc. and on further layers, the image can be identified. After combining all the layers into a particular layer, we can identify whether it is a person (or) object (or) animal. The CNN becomes more complex and identifies all of the image's components.

CNNs are a class of Deep Neural Networks that can recognize and classify particular features from images and are widely used for analyzing visual images. Their applications range from image and video recognition, image classification, medical image analysis, computer vision, and natural language processing. CNN has high accuracy, and because of the same, it is useful in image recognition. Image recognition has a wide range of uses in various industries such as medical image analysis, phone, security, recommendation systems, etc.

Three types of layers make up the CNN which are the convolutional layers, pooling layers, and fully-connected (FC) layers. When these layers are stacked, a CNN architecture will be formed. In addition to these three layers, there are two more important parameters which are the dropout layer and the activation function

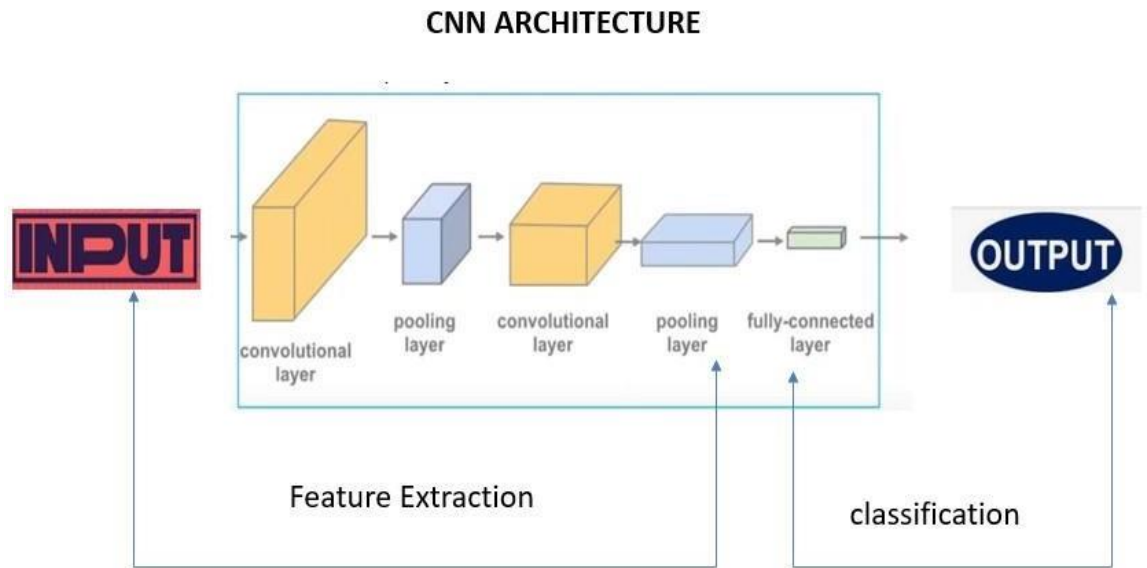


FIG 3.1 CONVOLUTIONAL NEURAL NETWORK

Types of CNN layers

a. Convolutional Layer

This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size $M \times M$. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image concerning the size of the filter ($M \times M$). The output is termed the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image. The convolution layer in CNN passes the result to the next layer once applying the convolution operation in the input. Convolutional layers in CNN benefit a lot as they ensure the spatial relationship between the pixels is intact.

For identifying a grayscale or black and white image we have a formula that is $(n \times n \times 1) * (f \times f \times c) = (n - f + 1) * (n - f + 1) * c$, for this if we use c number of edges then we get c number of images in the output of the proposed work. When the operation is done on a colored image, we have different formula as $(n \times n \times 3) * (f \times f \times 3) = (n - f + 1) * (n - f + 1) * 1$, in this, for a colored image we are having three channels as red, green, and blue, for this image a 3-dimensional cube (a filter of

3 channels). In this after superimposing we will get a single image as an output.

For example, a size of 6x6 with different numbers in each pixel and also a filter of size 3x3 pixel is multiplied then the resultant matrix would be 4x4. The values of the 4x4 matrix are obtained by superimposing a 3x3 filter with a 6x6 pixel.

1	6	9	10	2	8
2	5	1	8	4	2
3	7	4	9	10	3
9	8	3	6	7	9
8	0	9	4	7	2
9	10	12	6	9	8

 $*$

1	0	-1
1	0	-1
1	0	-1

 $=$

-8	-9	-2	14
6	-3	-13	9
4	-4	-8	5
2	2	1	-3

FIG 3.2 CONVOLUTIONAL NETWORK

b. Pooling

Similar to the convolutional layer pooling is responsible for dimensionality reduction. For performing this pooling layer we need to fix a particular size from a matrix and a stride need to be taken with a particular amount then the output will be an image with the same size of the filter as well as stride. This is done to reduce the image size as well as computational cost.. There will be the same number of inputs as well as outputs. There are two types of pooling they are:

- i. Max pooling: This method is the most commonly used in this filter with the maximum value taken and send to the output matrix.

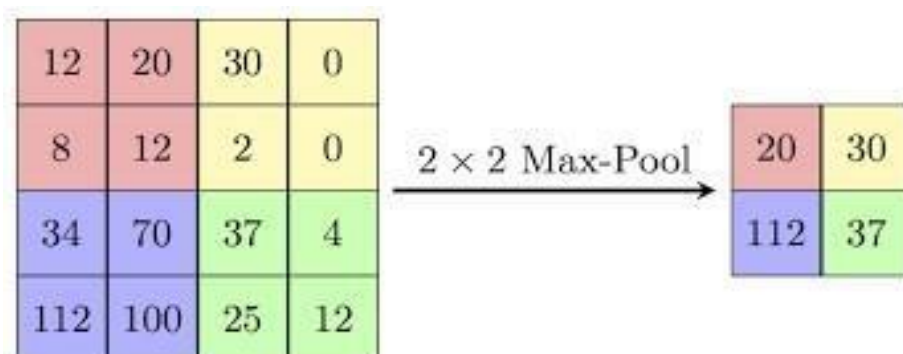


FIG 3.3 MAX POLING

- ii. Average pooling: This method is less preferred when compared with Max pooling. In this, the average of each filter is taken and sent to the output matrix.

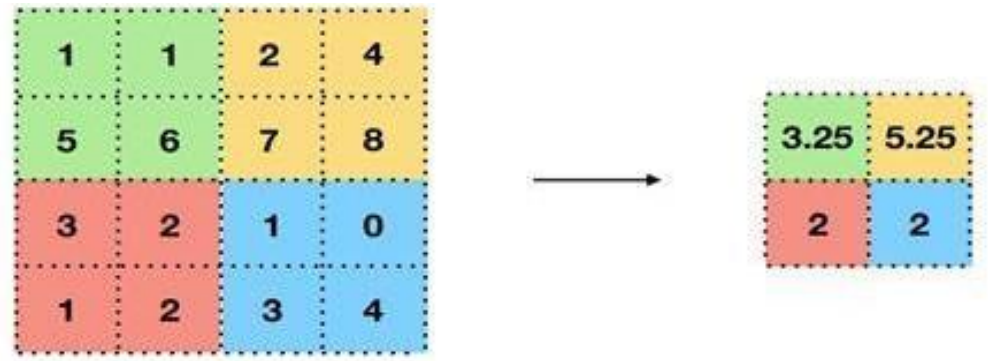


FIG 3.4 AVERAGE POLING

c. Activation function

The activation Function in the Neural Network shows the inputs of the node to the respective output

$$\text{Node Output} = \text{Activation (Sum of Weights of input data)}$$

i. Relu Activation Function

This changes the input to a maximum value of 0 or more. If the input is less than or equal to 0, the output will be 0 relu. If the value is more than 0, then the relu output will be same as the value.

$$\text{Relu}(x) = \max(0, x)$$

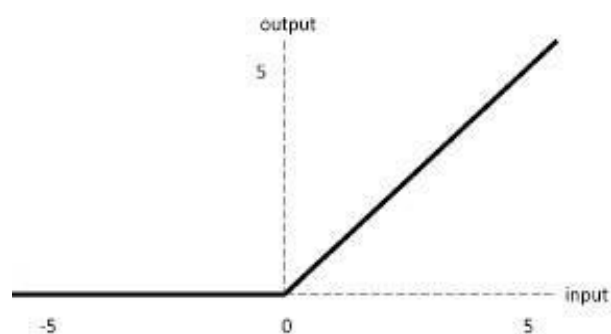


Fig 3.5: - GRAPH OF RELU ACTIVATION FUNCTION

ii. Softmax

It is an activation function which gives the probability values by comparing it with the trained values. If the value is positive, negative, or more than 1, then this activation function converts them into 0 and 1.

The softmax function is defined as follows:

For a vector of n real numbers $x = [x_1, x_2, \dots, x_n]$, the softmax function computes the probability distribution $y = [y_1, y_2, \dots, y_n]$ such that

$$y_j = \exp(x_j) / (\exp(x_1) + \exp(x_2) + \dots + \exp(x_n))$$

where $\exp()$ is the exponential function.

The softmax function ensures that the output probabilities add up to 1, and that they are in the range $[0, 1]$. It is often used as the output activation function in neural networks for classification tasks, where the goal is to predict the probability of each possible class given the input data.

The softmax function is differentiable, which allows it to be used in gradient-based optimization algorithms such as stochastic gra

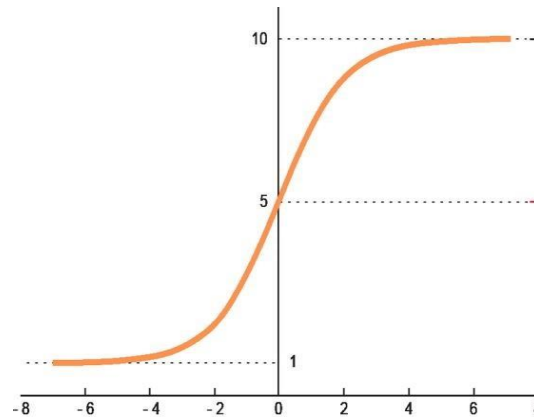


Fig 3.6: -GRAPH OF SOFTMAX ACTIVATION FUNCTION

d. Optimizer

It is a mechanism for changing the neural network characteristics such as weight and learning rate. The weight must be constantly updated during the training process to reach its required optimal values. The most widely used optimiser is Stochastic Gradient Descent. In our model we used the Adam which is a SGD optimizer type. It is a modified learning rate system which calculates every learning percentages for various variables. This helps for reducing overall loss and which results in the increase of accuracy .

Optimizers are an essential component of most machine learning algorithms, especially in deep learning where the models often have millions of parameters that need to be tuned. The choice of optimizer can have a significant impact on the performance of the model, both in terms of the accuracy of the predictions and the speed of training.

There are many different types of optimizers available, each with its own strengths and weaknesses. Some common optimizers used in machine learning include:

- 1) Stochastic Gradient Descent (SGD): A simple and widely used optimizer that updates the parameters of the model based on the gradient of the loss function with respect to the parameters.
- 2) Adam: A popular optimizer that combines the benefits of adaptive learning rates and momentum.
- 3) Adagrad: An optimizer that adapts the learning rate of each parameter based on the historical gradients for that parameter.
- 4) RMSprop: An optimizer that uses a moving average of the squared gradients to adjust the learning rate for each parameter.
- 5) Adadelta: An optimizer that uses a moving average of the gradients and the squared gradients to adapt the learning rate for each parameter.

The choice of optimizer depends on various factors such as the complexity of the model, the size of the dataset, and the availability of computational resources. It is often a matter of experimentation and fine-tuning to determine the best optimizer for a particular task.

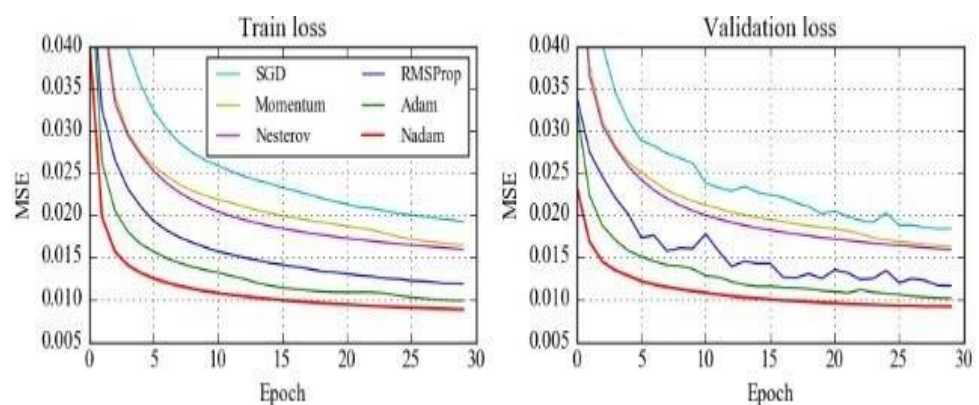


Fig 3.7: -GRAPH OF THE ADAM OPTIMIZER

e. FULLY CONNECTED LAYER:

The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture. In this, the input image from the previous layers is flattened and fed to the FC layer. The flattened vector then undergoes a few more FC layers where the mathematical function's operations usually take place. In this stage, the classification process begins to take place. The reason two layers are connected is that two fully connected layers will perform better than a single connected layer. These layers in CNN reduce human supervision.

Fully connected layers are used to classify the features. It will decide whether the image present in it is a human or animal or an object. The fully connected layer is a dense network of neurons connected to every neuron in the next layer as well as the previous layer. After the extraction of features from the convolutional layer and pooling layer, the final features are obtained as output and these features are created as a one-dimensional array. As there will be three fully connected layers, the one-dimensional array is taken as an input and passes through them and when it reaches the third fully connected layer it uses the SoftMax activation function to classify the image into a particular category.

3.3.4 PARAMETERS IN CNN MODEL:

a. Dropout

Usually, when all the features are connected to the FC layer, it can cause overfitting in the training dataset. Overfitting occurs when a particular model works so well on the training data harming the model's performance when used on new data. To overcome this problem, a dropout layer is utilized wherein a few neurons are dropped from the neural network during the training process resulting in a reduced size of the model. On passing a dropout of 0.3, 30% of the nodes are dropped out randomly from the neural network. Dropout results in improving the

performance of a machine learning model as it prevents overfitting by making the network simpler.

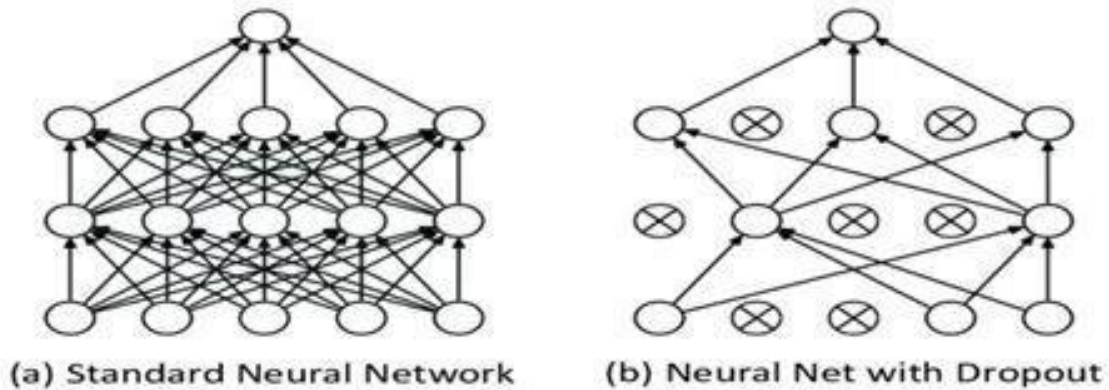


Fig 3.8: - DROPOUT LAYER

b. Activation Functions

Finally, one of the most important parameters of the CNN model is the activation function. They are used to learn and approximate any kind of continuous and complex relationship between variables of the network. In simple words, it decides which information of the model should fire in the forward direction and which ones should not at the end of the network. It adds non-linearity to the network. There are several commonly used activation functions such as the ReLU, Softmax, tanH, and Sigmoid functions. Each of these functions has a specific usage. For a binary classification CNN model, sigmoid and softmax functions are preferred and for multi-class classification, generally, softmax is used. In simple terms, activation functions in a CNN model determine whether a neuron should be activated or not.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

Looking at the disadvantages of all the above methodologies used in previous systems, the most common point that pops up is most of these systems implemented this problem statement using only a pre-defined dataset of faces with closed eyes and opened eyes. Also they had only visual types of alerting system to inform the driver of his state, which is not an effective alarm because visual alarms require one to be alert to see the alarm, which defeats the whole purpose at hand. Also, some of these systems' response time between finding the state of the driver and alerting the driver of his state was found to be too long to prevent mishaps in time. Some systems were found to be too sensitive to the eye blinks and yawns and other systems gave alarms continuously for a long time resulting in spamming the system. Our system aims to overcome all these issues with the previously existing systems and address these issues while giving the best accuracy in the results. Our basic idea is to monitor the physical state of the driver while he's driving using a live camera. We are making use of facial parameters to track the retina in the eye of the beholder, in case of frequent eye blinks while the driver is tired and also keep track of their mouth movements in case of yawning. When our system detects either of these changes our model will immediately emit an alarm sound as loud as a siren alarm to immediately awaken the driver from his poor state back to alertness.

4.1 SELECTED METHODOLOGY OR PROCESS MODEL

Drowsiness detection using Convolutional Neural Networks (CNNs) typically involves the following methodology:

- a) **Data Collection:** The first step is to collect data, such as images or videos, of drivers in various states of drowsiness. This data can be obtained through various sources, such as dashcam footage, simulator studies, or experiments.
- b) **Data Preprocessing:** The collected data needs to be preprocessed to prepare it for use in the CNN. This involves tasks such as resizing the images, converting them to grayscale, and normalizing the pixel values.
- c) **Data Augmentation:** To increase the amount of training data and prevent overfitting, data augmentation techniques such as image rotation, flipping, and

scaling can be applied to generate new training examples.

- d) **Model Architecture:** The CNN model needs to be designed to extract features from the input data and classify it into different drowsiness levels. The architecture typically consists of multiple convolutional layers, pooling layers, and fully connected layers.
- e) **Model Training:** The model is trained on the preprocessed data using techniques such as backpropagation and stochastic gradient descent. The model's performance is evaluated on a validation set, and hyperparameters such as learning rate and batch size are tuned to optimize the model's accuracy.
- f) **Model Testing:** The trained model is evaluated on a test set to measure its performance on unseen data. Metrics such as accuracy, precision, recall, and F1 score are used to evaluate the model's performance.
- g) **Deployment:** Once the model has been trained and tested, it can be deployed in real-world applications, such as in-vehicle monitoring systems or smartphone apps, to detect drowsiness in drivers and alert them to take a break or switch drivers.

Overall, this methodology provides a systematic approach to developing a CNN-based drowsiness detection system that can accurately and reliably detect drowsiness in drivers.

4.2 ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM

The block diagram of the proposed system has been shown in the below figures.

The camera captures the image of the person inside the car and sends that to the HOG model to train and it detects each feature from the face using facial landmark technique in the system. The next step in the process would be as it starts to find the position and condition of each feature to analyze, it should detect whether the person is sleeping or not. If any of the features especially the eye and the head pose/state of the person are detected to be ab normal then automatically the system start stop produce the siren sound. This is where we will get the final judgment of the state of the driver, whether he is concentrated or distracted.

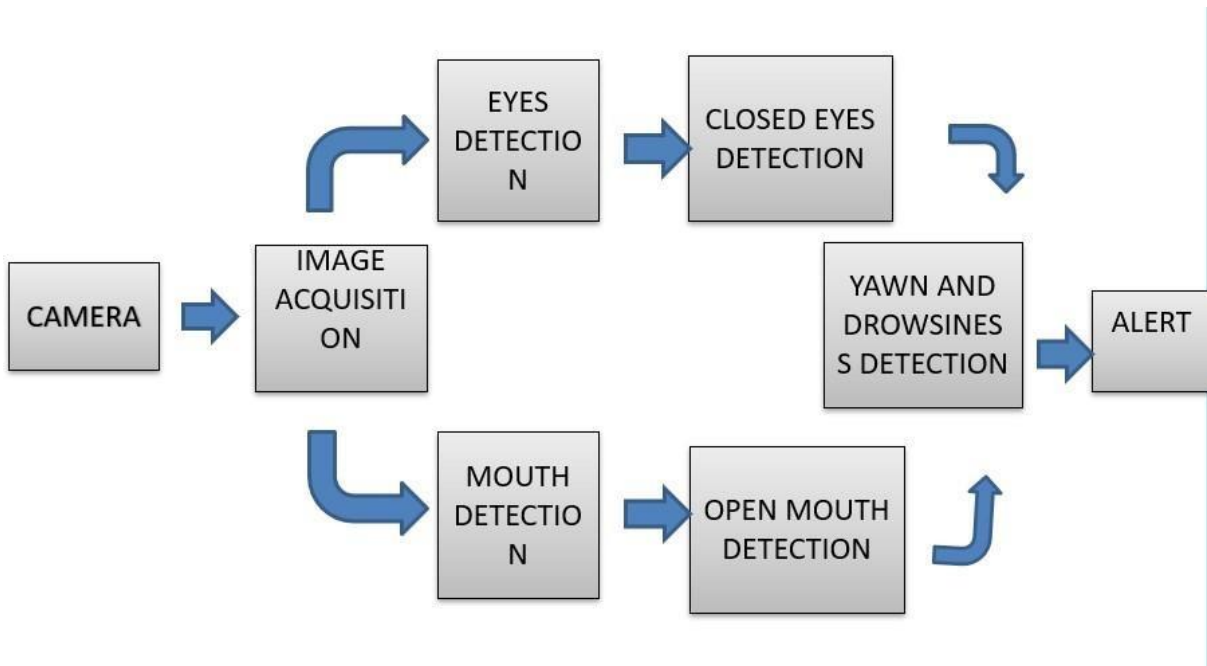


Fig 4.1 SYSTEM ARCHITECTURE

4.3 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED MODEL/SYSTEM

ANACONDA NAVIGATOR:

Anaconda Navigator is remembered for the Anaconda distribution and permits clients to send off applications and oversee condo packages, environments, and channels without using command-line commands. Navigator can search for packages, install them in an environment, run the packages and update them. With the help of the anaconda command prompt, all necessary packages are installed.

GIT-HUB:

Git-hub is used for cloning repositories like UCI and uploading the files into Git-hub and saving them there. GitHub also supports the development of open-source software, which is software with source code that anyone can inspect, modify, and enhance the models. Kaggle might also use for training the model as it supports faster execution as it supports GPU.

OPEN CV:

OpenCV (Open Source Computer Vision Library) is a popular open-source computer vision and machine learning library. It provides a wide range of tools and functions for image and video processing, feature detection, object recognition, and machine learning.

OpenCV can be used with various programming languages, including Python, C++, Java, and MATLAB. It supports different platforms, including Windows, Linux, Android, and macOS.

Some of the key features of OpenCV include:

- a) Image and Video Processing: OpenCV provides tools for loading, manipulating, and saving images and videos. It also supports different image and video file formats.
- b) Feature Detection: OpenCV includes functions for detecting and extracting features from images and videos, such as corners, edges, and blobs.
- c) Object Recognition: OpenCV includes functions for detecting and recognizing objects in images and videos using machine learning algorithms, such as Haar cascades and deep learning models.
- d) Machine Learning: OpenCV includes tools and functions for training and evaluating machine learning models, such as support vector machines (SVMs) and artificial neural networks (ANNs).
- e) GUI and Visualization: OpenCV provides functions for creating GUI applications and visualizing data, such as plotting and displaying images and videos.

OpenCV is widely used in various applications, including robotics, autonomous vehicles, medical imaging, security systems, and video surveillance. It has a large community of users and developers, and there are many resources available for learning and using OpenCV, such as documentation, tutorials, and sample code.

4.4 PROJECT MANAGEMENT PLAN

A project management plan can be structured as follows:

Weeks 1-2	<ul style="list-style-type: none">• Conduct research on existing drowsiness detection methods and CNN algorithms.
Weeks 2-4	<ul style="list-style-type: none">• Design and develop the CNN algorithm for drowsiness and yawning detection.
Weeks 4-8	<ul style="list-style-type: none">• Implement the CNN algorithm on the Raspberry Pi platform and integrate it with the infrared camera
Weeks 8-12	<ul style="list-style-type: none">• Train the CNN algorithm using the dataset of drivers exhibiting drowsiness.
Weeks 12-15	<ul style="list-style-type: none">• Validate the system's accuracy in detecting drowsiness using the separate dataset of drivers.• Finalize the system's design and prepare the user manual.

Project Scope:

- a) Develop a CNN algorithm for drowsiness detection.
- b) Implement the algorithm on a Raspberry Pi hardware platform.
- c) Integrate the system with a camera for real-time video input.
- d) Train the CNN algorithm on a dataset of drivers exhibiting drowsiness.
- e) Validate the system's accuracy in detecting drowsiness using a separate dataset of drivers.
- f) Provide a visual and audible warning to the driver if drowsiness is detected.

Project Deliverables:

- a) CNN algorithm for drowsiness detection
- b) Raspberry Pi-based hardware platform with integrated camera
- c) Trained CNN model
- d) Dataset of drivers exhibiting drowsiness
- e) Dataset of drivers for system validation
- f) Validation report
- g) User manual

4.5 FINANCIAL REPORT ON ESTIMATED COSTING

- Scope of the project: The scope of the project may vary based on the features and functionalities included in the system. More complex and advanced features may require additional resources and therefore increase the financial cost of the project.
- Technology stack: The choice of technology stack may impact the financial cost of the project. Some technologies may be more expensive than others, or may require additional licenses or subscriptions.
- Timeframe: The timeframe for completing the project may also impact the financial cost. A shorter timeframe may require additional resources to be allocated, leading to higher costs.
- External factors: External factors such as changes in regulations, market conditions, or unexpected events such as the COVID-19 pandemic may impact the financial cost of the project. These factors may require additional resources to be allocated or may cause delays, leading to additional costs.
- Personnel cost: This includes the salaries and benefits of the developers, project manager, and other team members involved in the project.
- Equipment cost: This includes the cost of hardware, software, and other equipment required for the development, testing, and deployment of the project.
- Data acquisition cost: This includes the cost of acquiring and processing data related to crop recommendations, fertilizer recommendations, and disease detection.

- Travel cost: This includes the cost of travel for team members to meet with farmers, agricultural experts, and other stakeholders to gather requirements and feedback.
- Miscellaneous cost: This includes other expenses such as office rent, legal fees, marketing and promotion, and other overhead costs.

Overall, the financial cost of the project will depend on the specific requirements and circumstances of the project.

4.6 TRANSITION/ SOFTWARE TO OPERATIONS PLAN

A transition/ software to operations plan for drowsiness detection system involves the steps that need to be taken to move the software from the development stage to an operational environment. Here is an example of a high-level plan that could be used for a drowsiness detection system:

- a) Define the Operational Environment: The first step is to define the operational environment where the drowsiness detection system will be deployed. This includes identifying the hardware, software, and networking infrastructure needed to run the system in a real-world setting.
- b) Develop the Deployment Strategy: Once the operational environment has been defined, the next step is to develop a deployment strategy. This involves defining the deployment process, identifying the necessary resources, and establishing timelines and milestones.
- c) Implement the System: After the deployment strategy has been established, the drowsiness detection system can be implemented in the operational environment. This includes installing the software, configuring the hardware and network infrastructure, and testing the system to ensure that it is functioning properly.
- d) Establish Support and Maintenance Procedures: Once the system has been deployed, support and maintenance procedures need to be established to ensure that the system remains operational over time. This includes establishing procedures for monitoring the system, performing upgrades and updates, and troubleshooting issues as they arise.
- e) Train End Users: Finally, end-users need to be trained on how to use the drowsiness detection system effectively. This includes providing training on the

software interface, how to interpret the results, and any necessary safety protocols.

Every software project needs a transition strategy because it entails handing off the system from the development team to the operations team, which is in charge of its upkeep and support.

The operations team will receive instruction and knowledge transfer from the development team, ensuring they have the skills and information required to support and maintain the system. A user manual outlining the system's features, functions, and operational needs will be made available.

The system will be deployed in stages, with testing and validation carried out at each step to make sure it's working correctly and effectively. The deployment will be carefully watched to make sure that any problems are dealt with right away.

In summary, a transition/ software to operations plan for a drowsiness detection system involves defining the operational environment, developing a deployment strategy, implementing the system, establishing support and maintenance procedures, and training end-users. By following these steps, the system can be effectively moved from the development stage to an operational environment, where it can be used to enhance driver safety.

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 DEVELOPMENT AND DEPLOYMENT SETUP

Generally, several phases are involved in the development stage such as research, data collection, preparing data, model development, training a model, evaluation the model and making prediction.

- During the research phase, a thorough investigation of existing literature and techniques related to Dog breed identification using CNNs will be conducted by the project team. This will involve reviewing academic papers, online resources, journals some IEEE conferences to gain a deep understanding of the topic and identify potential approaches.
- Collecting Data: Collecting the data is the most important thing for any model. It is of the utmost importance to collect reliable data so that model can find the correct patterns. The quality of the data that you feed to the machine will determine how accurate your model is. If you have incorrect or outdated data, you will have wrong outcomes or predictions which are not relevant.
- Preparing Data: Putting together all the data you have and randomizing it. This helps make sure that data is evenly distributed, and the ordering does not affect the learning process. Cleaning the data to remove unwanted data, missing values, rows, and columns, duplicate values, data type conversion, etc. You might even have to restructure the dataset and change the rows and columns or index of rows and columns. Visualizing the data to understand the structure as well as the model.
- Model development and training the model: In the model development and training phase, the architecture of the CNN model will be designed by the team and trained using the prepared dataset. This may involve experimenting with different model architectures and hyperparameters to achieve optimal performance. In the evaluation and validation phase, the model will be tested on a separate dataset to evaluate its performance and ensure that it is not overfitting. This may involve measuring metrics such as accuracy, precision,

recall, and F1 score.

- **Evaluation of the model:** After training your model, you have to check to see how it's performing. This is done by testing the performance of the model on previously unseen data. The unseen data used is the testing set that you split our data into earlier. If testing was done on the same data which is used for training, you will not get an accurate measure, as the model is already used to the data, and finds the same patterns in it, as it previously did. This will give you disproportionately high accuracy.
- **Making Prediction:** Need to make the prediction of project whether it is giving the correct prediction (or) not. If the prediction is wrong, we need to check the training data again until we get the correct prediction.

5.1.1 DATASET DESCRIPTION:

A drowsiness detection dataset typically consists of physiological and behavioral data collected from individuals during driving or other activities. The dataset usually includes recordings of brain activity, eye movement, and other physiological signals such as heart rate and skin conductance.

Additionally, the dataset may contain video or audio recordings of the individual's behavior during the activity, including head movements, yawning, and other signs of drowsiness. The data is typically labeled to indicate the level of drowsiness, ranging from alert to sleepy or fully asleep.

To train a drowsiness detection model, the dataset is split into training, validation, and test sets. The training set is used to train the model, while the validation set is used to tune the model's hyperparameters and prevent overfitting. Finally, the test set is used to evaluate the performance of the trained model.

The quality of a drowsiness detection dataset depends on several factors, including the quality of the data recording equipment, the number of participants, the diversity of activities, and the labeling accuracy. A high-quality dataset is essential for training accurate and reliable drowsiness detection models.

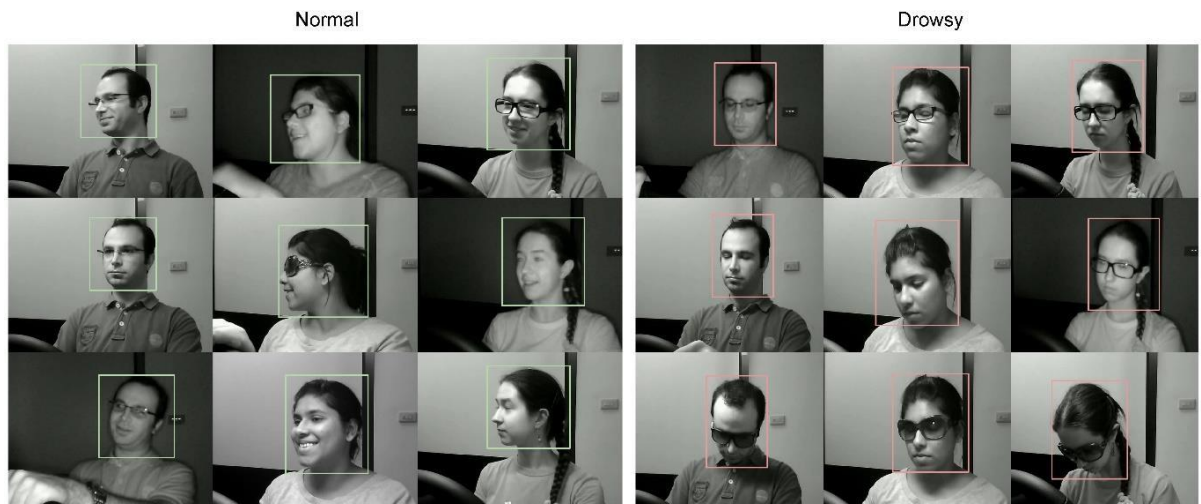


FIG 5.1 DROWSINESS DATASET



FIG 5.2 YAWNING DATASET

5.1.2 DATA PREPROCESSING:

- a) For drowsiness detection using a convolutional neural network (CNN), you can follow the following steps for data preprocessing:
- b) Data collection: Collect a dataset of recordings of individuals in various states of drowsiness, using sensors such as EEG, ECG, and eye-tracking devices.
- c) Data cleaning: Remove any outliers, missing data, or noise from the dataset.
- d) Data augmentation: Since CNNs require a large dataset for training, you can use data augmentation techniques like flipping, rotating, or adding noise to the existing data to create more samples.

- e) Feature extraction: Extract relevant features from the data, such as alpha and beta wave amplitudes from EEG signals, eyelid closure and pupil diameter from eye-tracking data.
- f) Normalization: Normalize the data to ensure that the features fall within the same range.
- g) Resizing: Resize the data to a fixed size, such as 256x256 pixels, to ensure that the input size is consistent for the CNN.
- h) Splitting the dataset: Divide the dataset into training, validation, and testing sets. The training set is used to train the model, the validation set is used to tune hyperparameters and prevent overfitting, while the testing set is used to evaluate the performance of the model.
- i) Labeling: Assign labels to the data, indicating whether the individual is alert or drowsy during the recording.

By following these steps, you can preprocess the data for drowsiness detection using a CNN.

5.2 ALGORITHMS:

CNN

A Convolutional Neural Network (CNN) is a type of deep neural network commonly used in image and video analysis, natural language processing, and speech recognition. It is inspired by the structure and function of the human visual cortex, which processes visual information in a hierarchical manner.

In a CNN, the input data (usually an image) is passed through a series of convolutional layers, where each layer applies a set of filters to the input to extract certain features from the image. The filters slide over the input data and perform element-wise multiplication and summation operations to produce a feature map. The feature map represents the presence of certain features in the input data.

After the convolutional layers, there may be one or more pooling layers, which downsample the feature maps by taking the maximum or average value in a certain area of the map. This reduces the spatial size of the feature maps and makes the network more efficient.

The output of the convolutional and pooling layers is then passed through one or more fully connected layers, which perform classification or regression tasks depending on the application. The fully connected layers take the flattened output of the previous layers and map it to a set of output classes or values.

CNNs have been shown to be very effective at image classification, object detection, and other computer vision tasks, and have achieved state-of-the-art results on many benchmark datasets.

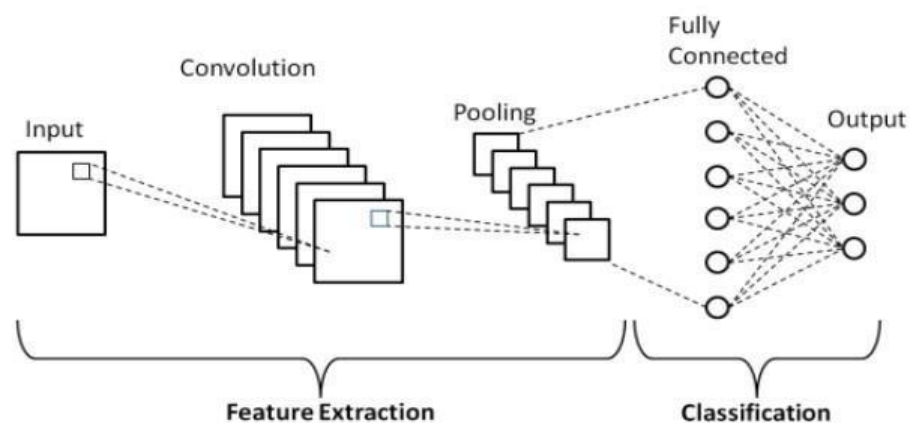


FIG 5.3 CONVOLUTIONAL NEURAL NETWORK

During training, the parameters of the filters in the convolutional layers are learned through backpropagation, which adjusts the weights of the neurons to minimize the error between the predicted output and the actual output.

CNNs have been shown to be highly effective at image recognition tasks, such as object detection, face recognition, and image classification, and are widely used in industries such as healthcare, automotive, and entertainment.

HAAR CASCADE ALGORITHM

Haar Cascade algorithm is a machine learning-based object detection algorithm used to identify objects in an image or video. It was proposed by Viola and Jones in their 2001 paper "Rapid Object Detection using a Boosted Cascade of Simple Features".

The algorithm uses Haar-like features, which are rectangular features that represent the difference in intensity between adjacent rectangular regions in an image. These features are simple and efficient to compute, and they can capture important information about the structure of an object.

The algorithm works by training a classifier using a large dataset of positive and negative examples of the object of interest. The positive examples are images containing the object, while the negative examples are images that do not contain the object. The classifier is trained using a boosting algorithm, which combines multiple weak classifiers into a strong classifier.

During the detection phase, the image is scanned with a sliding window of various sizes, and the Haar-like features are computed at each location. The classifier then evaluates each window as either containing or not containing the object. To improve efficiency, the algorithm uses a cascade of classifiers, where each classifier in the cascade is designed to quickly reject regions that are unlikely to contain the object.

Haar Cascade algorithm has been successfully applied in various applications such as face detection, pedestrian detection, and object tracking. It is widely used in computer vision and has been implemented in libraries such as OpenCV.

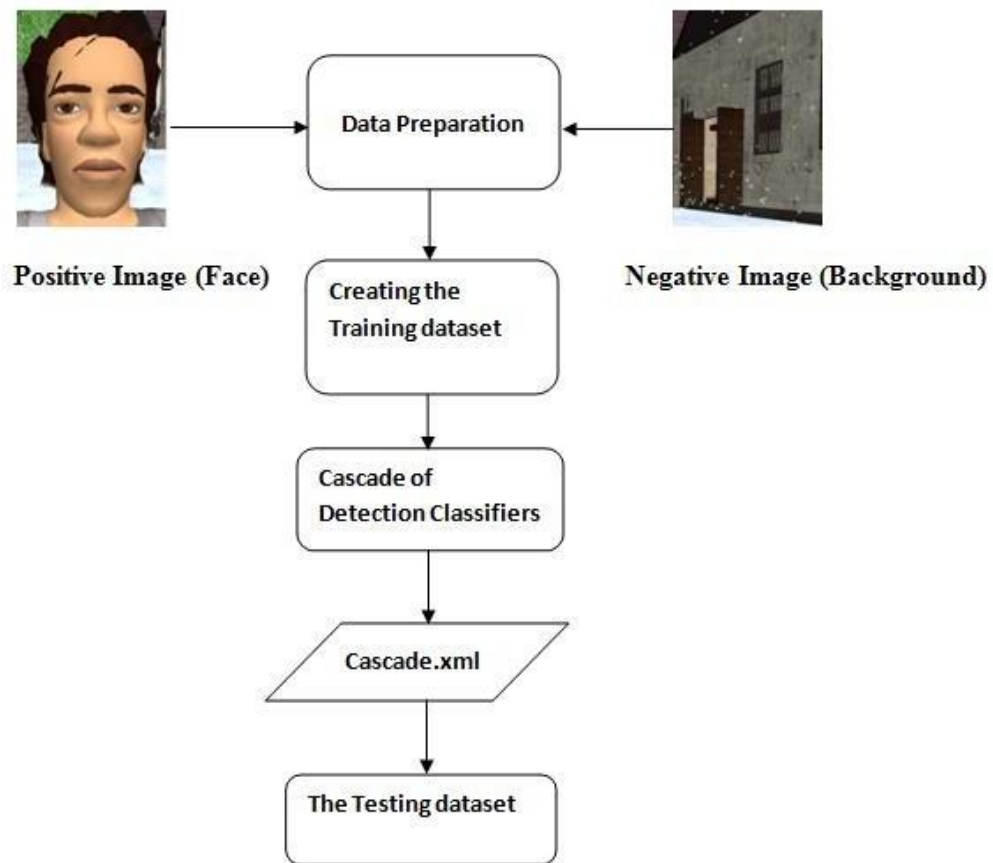


FIG 5.4 HAAR CASACADE ALGORITHM

5.3 LIBRARY

OPEN CV

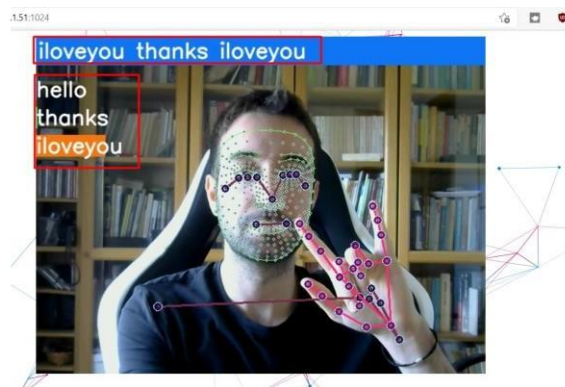


FIG 5.5 Open CV

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library designed to help developers create computer vision applications. It was first released in 2000 and has since become one of the most widely used computer vision libraries, with over 18 million downloads to date.

OpenCV provides a wide range of functions and algorithms for image and video processing, such as image filtering, feature detection, object recognition, and camera calibration. It also includes support for a variety of programming languages, including C++, Python, Java, and MATLAB.

Some of the key features of OpenCV include:

- a) Image and Video I/O: OpenCV can read and write images and video files in various formats, including BMP, JPEG, PNG, TIFF, and more.
- b) Image Processing: OpenCV includes a variety of image processing functions, such as image filtering, thresholding, morphology, and color space conversion.
- c) Feature Detection and Extraction: OpenCV provides algorithms for feature detection and extraction, such as Harris corner detection, SIFT, and SURF.
- d) Object Detection: OpenCV includes pre-trained models for object detection, such as Haar cascades and HOG+SVM
- e) Machine Learning: OpenCV also includes machine learning algorithms, such as support vector machines (SVM), k-nearest neighbors (k-NN), and neural networks.

OpenCV is widely used in many fields, including robotics, augmented reality, autonomous vehicles, and medical imaging. Its open-source nature and vast community support have made it a popular choice for computer vision researchers and developers.

CHAPTER 6

RESULTS AND DISCUSSION

Drowsiness detection using Convolutional Neural Networks (CNNs) has been an active area of research in recent years, with the potential to improve safety in various fields such as transportation and healthcare. In this task, the goal is to develop a model that can accurately detect drowsiness in real-time by analyzing visual cues such as eye and head movements.

Several studies have demonstrated the effectiveness of CNN-based approaches for drowsiness detection. These approaches typically involve training a CNN on a large dataset of annotated videos or images, where the model learns to automatically extract relevant features from the input data. The model is then able to classify new input as either drowsy or alert based on these learned features.

One of the main advantages of CNN-based approaches is their ability to automatically learn relevant features from the input data, without requiring manual feature engineering. This makes CNNs particularly useful in scenarios where the relevant features are not well-understood or difficult to define. Additionally, CNNs are capable of processing large amounts of data quickly, making them well-suited for real-time applications.

However, there are also some challenges associated with CNN-based approaches for drowsiness detection. One key challenge is the need for large amounts of annotated data to train the model effectively. Collecting and annotating such data can be time-consuming and expensive, particularly if the data needs to be collected in real-world scenarios. Additionally, CNN-based models can be computationally intensive, which can limit their use in resource-constrained environments.

Despite these challenges, several studies have demonstrated promising results for CNN-based approaches to drowsiness detection. For example, a recent study used a CNN to detect drowsiness from eye movement data with an accuracy of 92.5%. Another study used a CNN to detect drowsiness from images of drivers' faces,

achieving an accuracy of 86.8%. These results suggest that CNN-based approaches have the potential to improve safety in various applications by accurately detecting drowsiness in real-time.

In conclusion, CNN-based approaches have shown promising results for drowsiness detection in recent years. While there are some challenges associated with these approaches, their ability to automatically learn relevant features from the input data makes them well-suited for real-time applications. With continued research and development, CNN-based approaches to drowsiness detection have the potential to improve safety in a variety of fields.

Accuracy and loss for 50 Epochs

```

9533
Epoch 45/50
43/43 [=====] - 146s 3s/step - loss: 0.0900 - accuracy: 0.9636 - val_loss: 0.1229 - val_accuracy: 0.
9533
Epoch 46/50
43/43 [=====] - 151s 4s/step - loss: 0.0862 - accuracy: 0.9703 - val_loss: 0.1422 - val_accuracy: 0.
9619
Epoch 47/50
43/43 [=====] - 145s 3s/step - loss: 0.1127 - accuracy: 0.9614 - val_loss: 0.0959 - val_accuracy: 0.
9654
Epoch 48/50
43/43 [=====] - 147s 3s/step - loss: 0.1045 - accuracy: 0.9629 - val_loss: 0.0986 - val_accuracy: 0.
9689
Epoch 49/50
43/43 [=====] - 147s 3s/step - loss: 0.0785 - accuracy: 0.9755 - val_loss: 0.0893 - val_accuracy: 0.
9723
Epoch 50/50
43/43 [=====] - 159s 4s/step - loss: 0.0771 - accuracy: 0.9659 - val_loss: 0.0964 - val_accuracy: 0.
9689

```

FIG 6.1 ACCURACY OF EPOCHS

Predictions for test dataset

- 0 – Yawn
- 1 – No_yawn
- 2 – Closed
- 3 - Opened

```

In [22]: testpredict
Out[22]: array([[3, 3, 2, 3, 1, 3, 3, 2, 3, 2, 3, 3, 3, 2, 2, 2, 3, 3, 0, 3, 3, 3,
2, 3, 3, 2, 2, 2, 2, 2, 3, 3, 3, 2, 0, 3, 3, 2, 3, 3, 2, 3,
2, 2, 3, 3, 3, 3, 3, 2, 1, 3, 3, 3, 2, 0, 3, 3, 2, 2, 0, 3,
3, 3, 2, 2, 1, 3, 3, 3, 2, 0, 2, 3, 3, 3, 0, 3, 2, 2, 2, 1,
3, 2, 2, 3, 3, 3, 1, 2, 3, 0, 3, 3, 3, 3, 3, 3, 2, 3, 0, 2,
3, 1, 2, 3, 0, 3, 3, 3, 2, 2, 3, 3, 3, 1, 1, 2, 3, 3, 2, 0, 3,
0, 3, 0, 2, 3, 3, 2, 3, 2, 2, 3, 3, 3, 2, 3, 3, 0, 3, 3, 2,
3, 1, 0, 3, 0, 3, 2, 2, 3, 2, 3, 2, 3, 2, 3, 1, 2, 0, 3,
2, 3, 0, 2, 3, 0, 2, 0, 3, 2, 3, 1, 2, 1, 3, 2, 2, 2, 2, 0,
3, 3, 3, 0, 2, 0, 0, 1, 2, 3, 2, 0, 3, 1, 2, 2, 2, 2, 0, 3,
1, 2, 2, 3, 3, 3, 3, 0, 1, 2, 1, 0, 3, 3, 0, 3, 2, 3, 3, 0,
0, 0, 3, 3, 3, 3, 2, 3, 3, 2, 3, 2, 0, 2, 2, 2, 3, 2, 2,
3, 2, 0, 2, 3, 2, 2, 3, 3, 3, 3, 0, 0, 3, 3, 3, 3, 2,
2, 1, 2, 2, 2, 3, 2, 0, 2, 2, 1, 3, 2, 3, 1, 3, 3, 0, 3,
3, 2, 2, 1, 0, 0, 3, 3, 3, 3, 1, 3, 3, 2, 2, 1, 3, 3, 2,
3, 2, 3, 3, 3, 3, 2, 1, 3, 2, 2, 1, 3, 3, 3, 3, 2, 2, 2,
2, 2, 3, 2, 2, 2, 1, 0, 2, 2, 2, 1, 3, 3, 3, 3, 2, 2, 2,
2, 3, 3, 3, 0, 2, 1, 3, 2, 2, 1, 0, 2, 2, 3, 2, 3, 2, 3,
3, 1, 0, 3, 0, 2, 3, 2, 2, 3, 3, 3, 2, 3, 3, 2, 3, 2, 3,
2, 3, 3, 2, 3, 1, 3, 2, 2, 3, 3, 2, 3, 2, 1, 1, 2, 2, 1,
3, 3, 2, 3, 3, 2, 2, 1, 2, 2, 2, 2, 3, 2, 2, 3, 3, 2, 1,
2, 2, 3, 3, 0, 3, 3, 3, 2, 2, 2, 2, 2, 2, 3, 1, 2, 0,
2, 2, 3, 2, 3, 2, 2, 3, 2, 3, 2, 3, 2, 1, 0, 2, 2, 3, 0,
3, 3, 0, 1, 3, 2, 2, 2, 2, 2, 3, 2, 2, 3, 3, 3, 3, 2, 2,
2, 2, 3, 2, 2, 3, 3, 3, 1, 2, 1, 3, 3, 1, 0, 3, 3, 2, 0,
3, 2, 2, 2, 2, 2, 3, 2, 2, 2, 1, 3, 2, 3, 3, 3, 2, 3,
3, 2, 2, 3, 2, 2, 2], dtype=int64)

```

FIG 6.2 TESTING OF DATA SET

Graph of training and validation accuracies

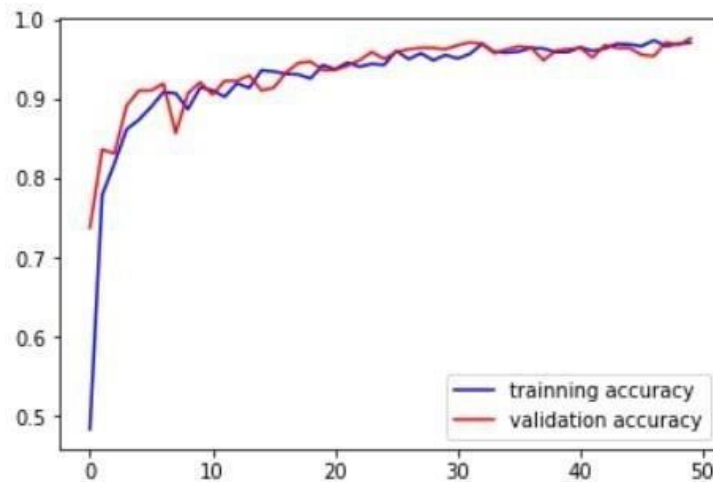


FIG 6.3 GRAPHY OF ACCURACY

Graph of training and validation losses

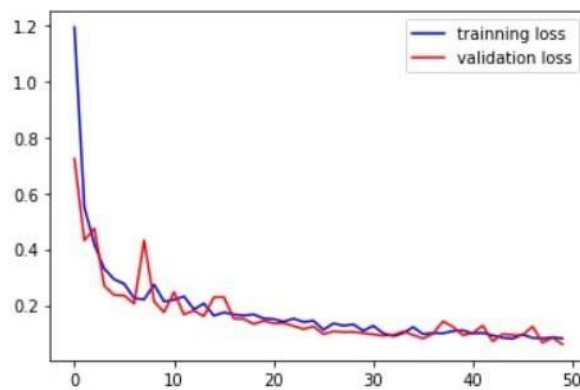


FIG 6.4 GRAPH OF LOSS

Live Detected Images



FIG 6.5 OUTPUT OF ACTIVE STATUS

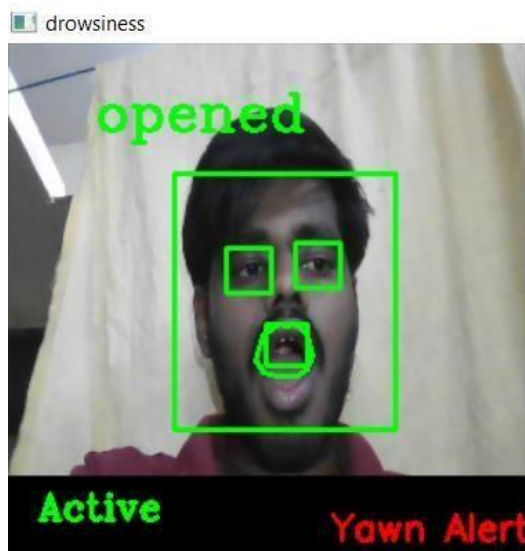


FIG 6.6 OUTPUT OF YAWN ALERT



FIG 6.7 OUTPUT OF DROWSY STATUS



FIG 6.8 NORMAL DRIVER OUTPUT



FIG 6.9 DROWSY DRIVER OUTPUT

ACCURATE YAWNING AND EYE LASHES DETECTION

SNO.	Absolute Positive Rating(Eye)	Absolute Negative Rating(Eye)	Inaccurate Positive Rate(Mouth)	Inaccurate Negative Rating(Mouth)
1	32.2 %	96.3 %	50.5 %	63.1 %
2	92.6 %	97.9 %	96.5 %	96.2 %
3	75.4 %	77.1 %	77.1 %	69.5 %
4	99.2 %	96.9 %	96.9 %	97.0 %
Accuracy	97.0 %	91.7 %	91.7 %	94.5 %

CHAPTER 7

CONCLUSION

7.1 CONCLUSION

In conclusion, the use of Convolutional Neural Networks (CNN) for drowsiness detection has shown promising results. CNN models can effectively extract relevant features from facial images, such as eye closure and head position, to predict whether an individual is drowsy or not.

However, the performance of CNN models can be influenced by various factors such as lighting conditions, camera quality, and variability in facial expressions. To overcome these challenges, researchers have explored various techniques such as data augmentation and transfer learning.

Overall, CNN-based drowsiness detection systems have the potential to improve safety in various industries such as transportation and healthcare by providing real-time monitoring of drowsiness in individuals. Future research can focus on improving the accuracy and reliability of these systems under various environmental conditions.

7.2 FUTURE WORK

There are several potential future directions for drowsiness detection using convolutional neural networks (CNNs). Here are a few ideas:

- a) Incorporating temporal information: Currently, most CNN-based approaches for drowsiness detection use a single image or frame as input. However, drowsiness is a temporal phenomenon, and it would be useful to incorporate temporal information into the model. This could be done by using multiple frames or by using a recurrent neural network (RNN) to model the temporal dynamics.
- b) Exploring new CNN architectures: While CNNs have been very successful in many computer vision tasks, there may be more specialized architectures that could be better suited for drowsiness detection. For example, recent work has

explored the use of attention mechanisms in CNNs, which could be used to focus on regions of the image that are most informative for drowsiness detection.

- c) Developing new datasets: While there are several datasets available for drowsiness detection, they are relatively small and may not fully capture the diversity of real-world scenarios. Developing new datasets with a wider range of subjects, lighting conditions, and driving scenarios could help to improve the robustness of CNN-based drowsiness detection models.
- d) Multi-modal approaches: While vision-based approaches have been successful for drowsiness detection, incorporating other modalities such as audio or physiological signals (e.g., heart rate, respiration) could improve performance. Multi-modal approaches could also help to mitigate the impact of environmental factors such as lighting conditions.
- e) Real-time implementation: While many CNN-based approaches for drowsiness detection have shown promising results, they are often computationally expensive and may not be suitable for real-time implementation on resource-constrained devices. Developing more efficient models that can be implemented in real-time could enable wider adoption of drowsiness detection technology in practical applications such as driver assistance systems.

7.3 RESEARCH ISSUES

Here are some research issues for drowsiness detection using convolutional neural networks (CNNs):

- a) Data imbalance: Drowsy driving events are relatively rare compared to non-drowsy events, resulting in imbalanced datasets. This can lead to biased models that perform poorly on the minority class. Developing techniques to handle imbalanced data is an important research issue for drowsiness detection using CNNs.
- b) Transferability: CNN models trained on one dataset may not perform well on another dataset due to differences in lighting conditions, camera angles, and other factors. Developing transfer learning techniques that enable CNNs to be trained on one dataset and applied to another dataset with different characteristics is an important research issue.

- c) **Generalizability:** While CNN-based drowsiness detection models have shown promising results, they are often evaluated on specific datasets and scenarios, and it is unclear how well they will perform in real-world settings. Developing techniques to evaluate and improve the generalizability of CNN-based models is an important research issue.
- d) **Robustness to environmental factors:** Environmental factors such as lighting conditions, weather, and road conditions can impact the performance of CNN-based drowsiness detection models. Developing techniques to improve the robustness of these models to environmental factors is an important research issue.
- e) **Interpretability:** While CNNs are powerful models for learning representations from data, they can be difficult to interpret. Developing techniques to enable researchers and practitioners to understand how CNN-based models are making decisions is an important research issue, especially for safety-critical applications such as driver assistance systems.

7.4 IMPLEMENTATION ISSUES

When implementing a drowsiness detection system using CNNs, there are several issues to consider:

- a) **Data quality:** The quality of the data used to train the CNN model is critical to the performance of the system. The training data should be diverse and representative of the target population. Additionally, the data should be labeled accurately, and any noise or outliers should be removed.
- b) **Model architecture:** The CNN model architecture should be carefully designed to balance accuracy and computational efficiency. The number of layers, filter sizes, and other hyperparameters should be optimized to achieve the desired performance while minimizing computation time.
- c) **Preprocessing:** Preprocessing the input data can improve the accuracy and robustness of the model. For example, normalizing the input images to a consistent scale and reducing image noise can improve the performance of the model.
- d) **Training time:** CNNs can be computationally expensive to train, particularly for large datasets or complex models. To mitigate this, techniques such as transfer learning or data augmentation can be used to reduce the amount of training

data required.

- e) Deployment: Once the model is trained, it must be deployed in a real-world system. This can introduce additional challenges related to computational efficiency, latency, and power consumption. For example, if the model is intended to run on a mobile device or embedded system, it must be optimized for low-power consumption and real-time performance.
- f) Ethical considerations: Drowsiness detection systems can have ethical implications, particularly if they are used in high-stakes scenarios such as driving. It is important to consider issues such as data privacy, bias, and transparency when designing and deploying these systems. Additionally, there may be legal and regulatory requirements that must be met before deploying a drowsiness detection system.

REFERENCES

- [1] Tiwari, K.S., Bhagat, S., Patil, N. and Nagare, P., 2019. IOT based driver drowsiness detection and health Monitoring System. *International Journal of Research and Analytical Reviews (IJRAR)*, 6(02), pp.163-167.
- [2] 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.
- [3] Hossain, M.Y. and George, F.P., 2018, October. IOT based real-time drowsy driving detection system for the prevention of road accidents. In *2018 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)* (Vol. 3, pp. 190-195). IEEE.
- [4] Sunagawa, M., Shikii, S.I., Nakai, W., Mochizuki, M., Kusakame, K. and Kitajima, H., 2019. Comprehensive drowsiness level detection model combining multimodal information. *IEEE Sensors Journal*, 20(7), pp.3709-3717.
- [5] Alkinani, M.H., Khan, W.Z. and Arshad, Q., 2020. Detecting human driver inattentive and aggressive driving behavior using deep learning: Recent advances, requirements and open challenges. *IEEE Access*, 8, pp.105008-105030.
- [6] Ngxande, M., Tapamo, J.R. and Burke, M., 2017. Driver drowsiness detection using behavioral measures and machine learning techniques: A review of state-of-art techniques. *2017 pattern recognition Association of South Africa and Robotics and mechatronics (PRASA-RobMech)*, pp.156-161.
- [7] Paulo, J.R., Pires, G. and Nunes, U.J., 2021. Cross-subject zero calibration driver's drowsiness detection: Exploring spatiotemporal image encoding of EEG signals for convolutional neural network classification. *IEEE transactions on neural systems and rehabilitation engineering*, 29, pp.905-915.
- [8] Karuppusamy, N.S. and Kang, B.Y., 2020. Multimodal system to detect driver fatigue using EEG, gyroscope, and image processing. *IEEE Access*, 8, pp.129645-129667.
- [9] Hu, Y., Lu, M., Xie, C. and Lu, X., 2019. Driver drowsiness recognition via 3D conditional GAN and two-level attention Bi-LSTM. *IEEE Transactions on Circuits*

and Systems for Video Technology, 30(12), pp.4755-4768.

- [10] Bala, U.C. and Sarath, T.V., 2020, June. Internet of things based intelligent drowsiness alert system. In *2020 5th international conference on communication and electronics systems (ICCES)* (pp. 594-598). IEEE.

APPENDIX

A. SOURCE CODE

```
# importing the necessary libraries
import numpy
as np import
pandas as pd
import os import
cv2
from tensorflow.keras.models import load_model
import tensorflow as tf
import matplotlib.pyplot as plt

os.chdir("C:\\Users\\abhis\\PycharmProjects\\MiniProject")

labels=os.listdir("train")

# creating an array for yawn and no yawn images from dataset with index
def yawn(direc="train",
face_cas_path="haarcascade_frontalface_default.xml"):    yaw_no = []
    IMG_SIZE = 145    categories =
["yawn", "no_yawn"]    for category in
categories:        path = os.path.join(direc,
category)        class_num1 =
categories.index(category)        for image
in os.listdir(path_link):
        image_array = cv2.imread(os.path.join(path_link, image), cv2.IMREAD_COLOR)
        face_cascade = cv2.CascadeClassifier(face_cas_path)
        faces = face_cascade.detectMultiScale(image_array, 1.3, 5)
        for (x, y, w, h) in faces:
            img = cv2.rectangle(image_array, (x, y), (x+w, y+h), (0, 255, 0), 2)
        roi_color = img[y:y+h, x:x+w]
        resized_array = cv2.resize(roi_color, (IMG_SIZE,
IMG_SIZE))        yaw_no.append([resized_array, class_num1])
return yaw_no

yawn_no_yawn = yawn()
# creating an array of open and closed eye images from dataset with index
def
get_data(dir_path="train",face_cas="haarcascade_frontalface_default.xml"
, eye_cas="haarcascade.xml"):    labels = ['Closed', 'Open']
    IMG_SIZE = 145
    data = []    for label in labels:
        path = os.path.join(dir_path, label)
        class_num 2= labels.index(label)
        class_num +=2        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 2])
            except Exception as e:
                print(e)
    return data
```

```

data_train =
get_data()

def append_data():
yaw_no = yawn()
data = get_data()
yaw_no.extend(data)
return np.array(yaw_no)

new_data = append_data()
X = [] y = [] for picture,
label in new_data:
X.append(picture)
y.append(label)

X = np.array(X)
X = X.reshape(-1, 145, 145, 3)

from sklearn.preprocessing import LabelBinarizer
label_bin = LabelBinarizer()
y = label_bin.fit_transform(y)

y = np.array(y)
from sklearn.model_selection import train_test_split

#Splitting the images in to train, valisdation and test set using data augmentation
X_train, X_test, y_train, y_test = train_test_split(X, y, 42, test_size=0.30)

# importing the necessary libraries for creating CNN model
from tensorflow.keras.layers import Input, Dense, Flatten, Conv2D, MaxPooling2D,
Dropout from tensorflow.keras.models import Model
from tensorflow.keras.models import Sequential from
keras.preprocessing.image import ImageDataGenerator
import tensorflow as tf

train_generator = ImageDataGenerator(rescale=1/255, zoom_range=0.2,
horizontal_flip=True, rotation_range=30)
test_generator = ImageDataGenerator(rescale=1/255)

train_generator = train_generator.flow(np.array(X_train), y_train, shuffle=False)
test_generator = test_generator.flow(np.array(X_test), y_test, shuffle=False)

# creating a CNN model
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()

# training the built CNN model
history = model.fit(train_generator, epochs=50, validation_data=test_generator,
shuffle=True, validation_steps=len(test_generator))

model.save("drowsiness_new6.h5")

testpredict = np.argmax(model.predict(X_test), axis=-1)
model=load_model('drowsiness_new6.h5')

testpredict          //it outputs values of predicted images
labels_new = ["yawn", "no_yawn", "Closed", "Open"]

# plotting graphs for accuracy and loss
accuracy = history.history['accuracy']
val_accuracy =
history.history['val_accuracy']
loss = history.history['loss']
val_loss =
history.history['val_loss'] epochs
= range(len(accuracy))

plt.plot(epochs, accuracy, "b", label="training accuracy")
plt.plot(epochs, val_accuracy, "r", label="validation
accuracy") plt.legend() plt.show() plt.plot(epochs, loss, "b",
label="training loss") plt.plot(epochs, val_loss, "r",
label="validation loss") plt.legend() plt.show()
from sklearn.metrics import classification_report
print(classification_report(np.argmax(y_test, axis=1), testpredict,
target_names=labels_new))

labels_new = ["yawn", "no_yawn", "Closed", "Open"]

IMG_SIZE = 145
def prepare(filepath, face_cas="haarcascade_frontalface_default.xml"):
    img_array = cv2.imread(filepath, cv2.IMREAD_COLOR)
    img_array = img_array / 255
    resized_array = cv2.resize(img_array, (IMG_SIZE,
IMG_SIZE))    return resized_array.reshape(-1, IMG_SIZE,
IMG_SIZE, 3) model =

```

```

tf.keras.models.load_model("drowsiness_new6.h5")

YAWN_THRESH = 20
def lip_distance(shape):
    top_lip = shape[50:53]
    top_lip = np.concatenate((top_lip, shape[61:64]))

    low_lip = shape[56:59]
    low_lip = np.concatenate((low_lip, shape[65:68]))

    top_mean = np.mean(top_lip, axis=0)
    low_mean = np.mean(low_lip, axis=0)

    distance = abs(top_mean[1] - low_mean[1])
    return distance
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

count=0 counter = 0
cap=cv2.VideoCapture(
0) while True:
    sucess,frame =
    cap.read()    try:
        frame = cv2.resize(frame,(300,
300))    except:        break
        gray=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
        eyes=eye_cascade.detectMultiScale(gray,1.1,4)

        rects = detector.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5,
minSize=(30,
30),flags=cv2.CASCADE_SCALE_IMAGE)
        for (x, y, w, h) in eyes:
            roi_gray=gray[y:y+h, x:x+w]
            roi_color=frame[y:y+h, x:x+w]
            cv2.rectangle(frame, (x, y), (x + w, y + h),
(0,255,0),2)
            eyess=eye_cascade.detectMultiScale(roi_gray)    if
            len(eyess)==0:        print("eyes are not detected")
            else:
                for(ex,ey,ew,eh) in eyess:
                    eyess_roi=roi_color[ey:ey+eh,ex:ex+ew]

            gray=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
            #faces = face_cascade.detectMultiScale(gray, 1.3,
3)    for (x, y, w, h) in rects:
                cv2.rectangle(frame, (x, y), (x + w , y + h + 20), (0, 255, 0), 2)
                rect = dlib.rectangle(int(x), int(y), int(x + w),int(y + h))

                shape = predictor(gray, rect)    shape =
                face_utils.shape_to_np(shape)    distance =
                lip_distance(shape)    lip = shape[48:60]
                cv2.drawContours(frame, [lip], -1, (0,255, 0), 2)

```

```

if (distance > YAWN_THRESH):
    x2,y2,w2,h2 = 175,250,125,75
    cv2.rectangle(frame,(x2,y2),(x2+w2,y2+h2),(0,0,0),-1)
    cv2.putText(frame,"Yawn Alert",(x2+int(w2/10),
    y2+int(h2/2)), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0,
    255), 2)

    cv2.putText(frame, "YAWN: {:.2f}".format(distance), (300, 60),
    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

font=cv2.FONT_HERSHEY_COMPLEX

    resized_img=cv2.resize(eyess_roi,(145,145))
    final_img=resized_img.reshape(-1,145,145,3)
    final_image=final_img/255
    predictions=np.argmax(model.predict(final_image))

    if(predictions==3):
        status = "opened"
        cv2.putText(frame,status,(50,50),font,1,(0,255,0),2,cv2.LINE_4)
        x1,y1,w1,h1=0,250,175,50

        cv2.rectangle(frame,(x1,y1),(x1+w1,y1+h1),(0,0,0),-1)

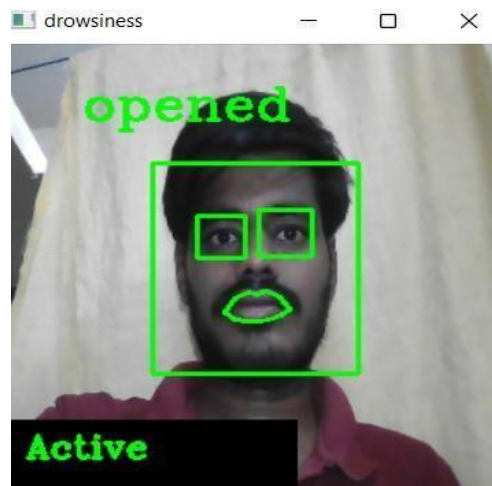
        cv2.putText(frame,'Active',(x1+int(w1/10),y1+int(h1/2)),cv2.FONT_HERSHEY_COMPLEX,0.7,(0,255,0),2)
    else:
        count=count+1
        status="closed "
        cv2.putText(frame,status,(50,50),font,1,(0,0,255),2,cv2.LINE_4)
        cv2.rectangle(frame, (x, y), (x + w , y + h + 20), (0, 255, 0), 2)
    if count>20:
        x1,y1,w1,h1=0,250,175,75
        cv2.rectangle(frame,(x1,y1),(x1+w1,y1+h1),(0,0,0),-1)

        cv2.putText(frame,'Drowsy',(x1+int(w1/10),y1+int(h1/2)),cv2.FONT_HERSHEY_COMPLEX,0.7,(0,0,255),2)
        while(count>=0):
            count = count - 1

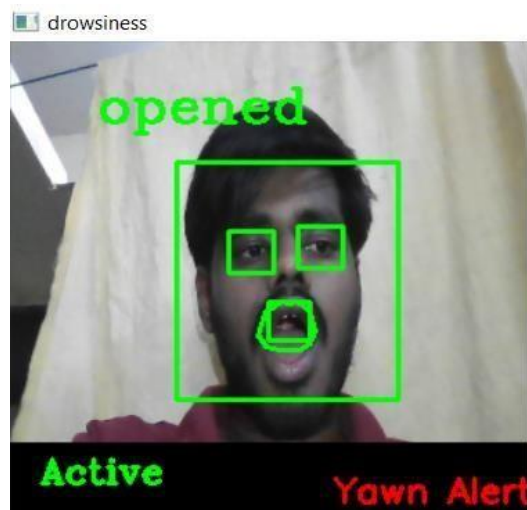
        cv2.putText(frame,'Drowsy',(x1+int(w1/10),y1+int(h1/2)),cv2.FONT_HERSHEY_COMPLEX,0.7,(0,0,255),2)
        cv2.imshow("drowsiness",frame)
        if cv2.waitKey(1) & 0xFF==ord('q'):
            break
cap.release()

```


B. SCREEN SHOTS



OUTPUT OF ACTIVE STATUS



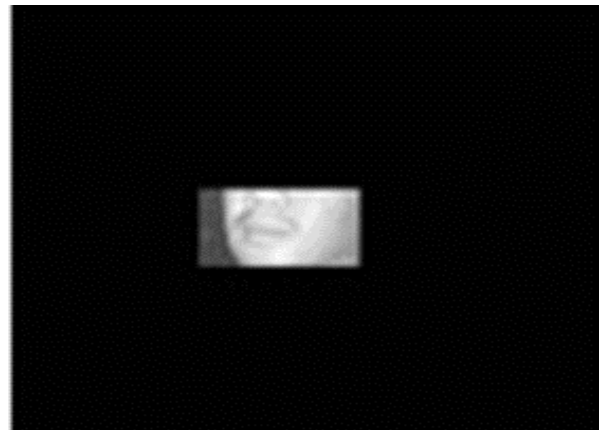
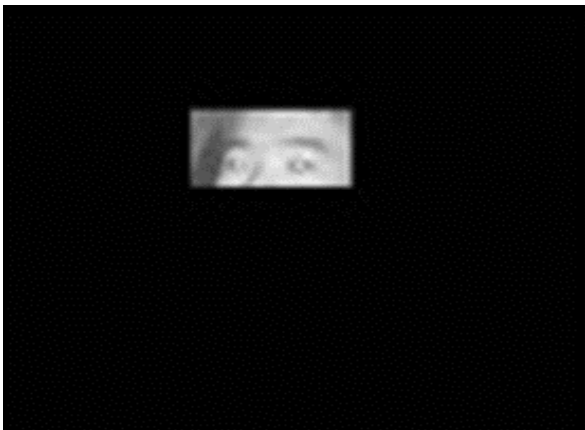
OUTPUT OF YAWN ALERT



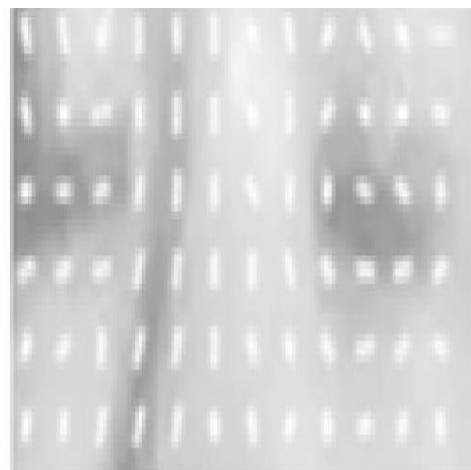
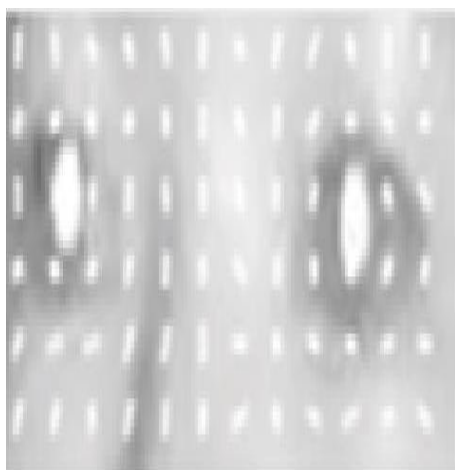
OUTPUT OF DROWSY STATUS



RASPBERRY PI MODULE



DETECTION OF EYES AND MOUTH IN HUMAN FACE



HOG OF EYES AND MOUTH



NORMAL DRIVER OUTPUT



DROWSY DRIVER OUTPUT

A METHOD DRIVER'S EYES CLOSURE AND YAWNING DETECTION BY INFRARED CAMERA

R. Subhash, UG student, Mail id: raavisubhash9492228024@gmail.com

S. Shanmukh praneeth, UG student, Mail id: praneethsappa9912@gmail.com

Dr. Kalaiarasi G, Assistant Professor, Mail id: kalaiarasi.cse@sathyabama.ac.in

Department of computer science and Engineering

Sathyabama Institute of Science and Technology – CHENNAI

ABSTRACT:

Diversions can result in hazardous incidents that are challenging to prevent as the world's population rises. Many studies have been conducted in an effort to figure out how to stop the death toll. The designers of the car put the safety of the occupants first. Airbags are made to increase security and save lives of occupants, but they do not prevent accidents from happening. The primary causes are general exhaustion and distraction from phone alerts. In the past, a wide range of methods based on behavioral measurements and machine learning approaches have been examined to identify driver distraction. These algorithms are necessary given the rapid expansion of such technology. The greatest frequent danger to their lives and the lives of others nowadays is driving while intoxicated. Although we cannot help someone stop being sleepy, we can make them awake and alert while they are driving. The suggested model was created by a few persons utilizing various technologies. But in order to increase the detection's precision, this model was created using specific technologies, namely OpenCV and Keras. A CNN model developed using Keras will be trained using the dataset after the driver's face is detected using OpenCV and a Haar Cascade Classifier. This trained model will output whether the driver is alert or tired based on the input of the detected face. Convolution neural networks are used in this study's new methodology to identify the distraction and immediately play a loud siren to provide a sensory shock that restores attentiveness. The aforementioned hybrid strategy would result in the greatest immediate resolution to such problems in the future when combined with appropriate instruction. A Convolutional Neural Network capable of identifying the eye and mouth detection in a given image is built, trained, and tested using Keras and TensorFlow. 2000 distinct images total, 2000 of which are yawning images, make up this dataset. These photos are loaded, then transformed into a NumPy array and normalized. To have the best precision, we accomplish this.

KEYWORDS: Convolutional Neural Networks, Keras, TensorFlow, NumPy, Pandas, Matplotlib, Raspberry Pi IOT device

I. INTRODUCTION

The increase in the demand for modern transportation in recent years necessitates faster advancements in automobile safety. People's primary mode of transportation right now is their car. Although it has changed people's lifestyles and made daily tasks more convenient, it has also been linked to a number of negative side effects, such as traffic jams and road accidents caused by preoccupation, fatigue, and exhaustion. These grave and hidden risks are to blame for several fatalities. Scientists have now been attempting to stop any more loss by anticipatorily identifying such signs far in advance. These methods of recognition are classified as both subjective and objective detection. The evaluation of the subjective detection technique, which the driver is obligated to take part in, is connected to the driver's subjective viewpoints through behaviours like self-questioning. The risk posed by cars driven

by fatigued drivers is then estimated using these data, helping those drivers make schedule adjustments. The objective detection approach, however, does not require their input because it continuously analyses their physiological state and driving behavior traits. The information gathered is used to estimate the amount of driver weariness. Moreover, touch and non-contact are the two categories for objective detection. As they only require sophisticated cameras and computer vision technologies, noncontact methods are less expensive and more practical than contact approaches, which enables widespread use of the device. For our issue statement, the noncontact method has been frequently adopted due to its simple installation and inexpensive cost. To determine a driver's state of fatigue, Attention Technologies and Smart Eye, for example, track the driver's eye movement and head posture. In this study, we suggest a non-contact way to gauge a driver's level of weariness. Our solution eliminates the need for the

driver to carry any on- or in-body equipment by relying just on the camera positioned on the vehicle. Each frame image in our design is analyzed and used to determine the driver's level of attention.

II. LITERATURE SURVEY

Researchers and automotive corporations have investigated this issue in-depth for the past five years in an effort to find a solution, and all of their approaches range from examining numerous distracting behavioral patterns to examining the driver's vital signs.

In addition to including an eye-blink monitoring device, Dr. K.S. Tiwari et al. work [1] also supplied a buzzer to inform the driver of his condition. In contrast, Ceerthi Bala et al. research [10] suggested a way to notify the traffic department when preoccupation is mistakenly observed.

Neurocognitive data, particularly from EEG, has been used in certain studies to indicate variations in brain dynamics when there is a change in attentiveness while driving [3]. He can comprehend the research conducted by Jap et al. on the use of EEG was used to detect drowsiness and tiredness, and it was discovered that when the subject—in this case, the driver—was preoccupied or impacted by fatigue, the ratio of slow to fast EEG waves rose. According to Gharagozlou et al. research [11], band power features and EEG signal entropy features can be used to evaluate additional levels of fatigue and distraction. These features demonstrate a noticeable rise in alpha power that is directly correlated with driver fatigue.

Rateb Jabbar et al. claim that using facial landmarks along with a convolution neural network could increase the efficacy of fatigue detection (CNN). J Hu and J Min's research integrated entropy features with a gradient boosting decision tree model. H. Zeng et al. developed a Residual Convolution Neural Network (EEGConv-R) with deep learning models for fatigue categorization using data collected from 10 healthy individuals across 16 channels.

A deep neural network and a support vector machine (SVM) classifier at the top layer were also suggested in P.P. San's research article. Intra-subject techniques form the foundation of other earlier efforts. According to H. Zeng et al., some cross-subject techniques also call for pooling EEG samples from all individuals before randomly dividing them into training and testing groups. Because of the approach's inherent randomness, some samples from training participants are mixed in with those from testing subjects, in which there is no cross-subject. In the study by Y. Li et al., the authors adjust the data distributions of the source and the target in order to increase the effectiveness of the classification in a cross-subject setting. Domain adaptation is a subset of transfer learning.

The eye closure ratio was suggested as the input parameter for a non-intrusive system by Md. Yousuf Hossain et al. In the

study by Y. Liu et al., EEG characteristics, statistics, higher. The classifiers logistic regression, linear discriminant analysis, 1 nearest neighbour, linear SVM, and Nave Bayes were used to combine order crossing, fractal dimension, signal energy, and spectral power.

A study challenge was put out by Wisaroot and Charoenpong in the area of recognising driver fatigue in poor light. In this post, we suggested a technique for detecting tiredness that involves utilising an infrared camera to spot the driver yawning and their eyes shutting. The four steps of this method are face detection, eye detection, mouth detection, and closed eye and yawning detection. In order to evaluate the performance of the suggested tactic, 3,760 pictures were employed. The outcomes of the trial demonstrated how effective the suggested approach was. The technique utilised in this study has the advantage of being able to identify yawning and eye closure in low-light conditions.

III. EXISTING WORK

Charoenpong and Wisaroot made the paper. The key ideas are to identify the driver's face and designate an area of interest for it (ROI). The next step is to identify mouth and ocular targets using ROI. Data from an infrared 2D camera is first collected, then processed in MATLAB R2015a. Image acquisition, face detection, eye detection, mouth detection, and eyes closed and yawning detection are the five processes in the flowchart. One technique that can recognize faces is the Haar-like feature. Michael Jones and Paul Viola are the authors of this technique. In this stage, an area of interest (ROI) is created in order to locate the driver's face. The dominant facial feature is classified using the Harr-like feature, which compares threshold and polarity to the difference between the shading rectangle and the regular rectangle

The author had to divide the face image into two half. This technique limits the ability to see the driver's eyes and mouth.

The driver's image is slightly skewed due to the camera settings, thus this stage rotates the image by around three degrees to the left and right in order to exploit the Haar-like characteristic to identify the driver's eyes and mouth.

Author usage of classify data machine that supports vectors (SVM). Normal photos must be converted into vector data by obtaining the Histogram of orientated images prior to training the SVM.

(HOG). HOG is a technique that divides a picture into several cells, gathers histograms from each gradient, and calculates the scale and vector of each cell [13]. The HOG of each pixel is a vector.

The following stage includes two groups of data, such as eyes open and closed. SVM comes in a variety of forms. SVM is appropriate for this paper. SVM's guiding principles are learning models for data analysis.

IV PROPOSED METHOD

The proposed technique focuses on understanding the layers and architecture of convolutional neural networks. As the CNN model involves various layer types, including the convolution Layer, pooling Layer, and fully connected Layer. Given that this CNN model is the most essential deep learning model for image categorization.

A CNN, also known as a convolutional neural network, is a fascinating subject with a wide range of practical applications, including self-driving cars, facial recognition, object detection, motion or movement detection, etc. Convolutional neural networks are positioned on top of neural networks, which are the fundamental building blocks of deep learning. Using filters is the convolutional neural network's primary principle. The filters are in charge of spotting photos or objects. These filters can also be used to identify the image's edges. These edges will be transmitted to the following layer so that it can identify the image's ears, eyes, nose, and other features. We may determine whether a layer is effective by integrating all the layers into that one layer.

i. CONVOLUTIONAL LAYER:

The advanced learning system CNN, which was created to process pixels, is utilized to recognize and process formats. The convolutional network requires less preprocessing than other classification algorithms. Although filters were created using more traditional techniques, CNN can examine these properties using its hidden layers.

The first layer of CNN is the convolutional layer. It serves as the main structural element and the brain of convolutional neural networks. The edges and features of the images are recognised by this layer. The success of CNN is primarily due to this layer. We employ numerous filters in a single convolutional layer, and various filters aid in the detection of various features in a picture, such as the detection of a circle, a horizontal edge, and so on.

CNN is made up primarily of two blocks.

1. Feature Extraction: It consists of three layers: the input layer, the convolutional layer, and the pooling layer.

2. Classification: It has a fully connected layer as well as an output layer.

In order to extract features from the image, this layer transforms data in two dimensions. The input data is convolved through filters throughout this procedure. This layer's filters consist of two-dimensional array matrices.

The amount of pre-processing used by CNNs and other image categorization methods is modest. The network learns to optimise the filters (or kernels) by autonomous learning, in contrast to conventional techniques, where these filters are manually constructed. This feature extraction's independence from previous knowledge and human interaction is a key

benefit.

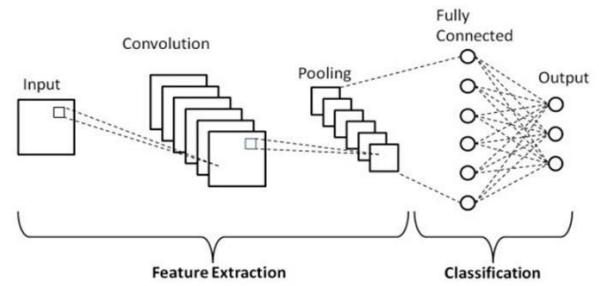


Fig 1.1: -Convolutional Neural Network

ii. MAX POOLING LAYER

Each CNN convolutional layers frequently include this kind of feature. By reducing the amount of pixels emitted from the convoluted layer, it reduces the size of images.

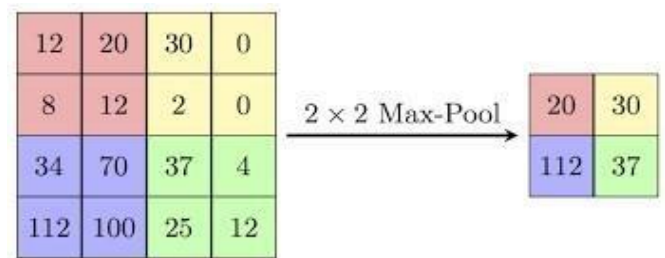


Fig 1.2: -Max Pooling process

iii. ACTIVATION FUNCTION

The neural network's activation function displays the inputs to each node's associated output.

Node Output = Activation (Sum of Weights of input data)

a. Relu Activation Function

This changes the input to a maximum value of 0 or more. If the input is less than or equal to 0, the output will be 0 relu. If the value is more than 0, then the relu output will be same as the value.

$$\text{Relu}(x) = \max(0, x)$$

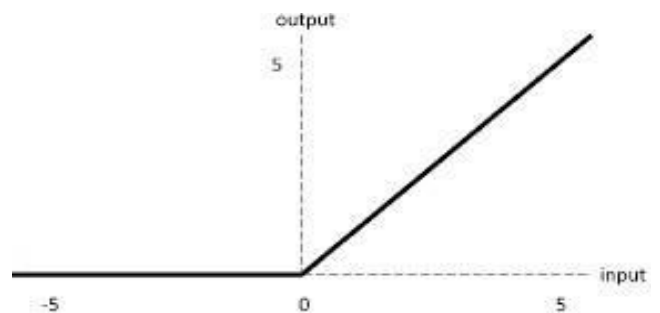


Fig 1.3: - Graph of Relu Activation Function

b. Soft Max

It is an activation function which gives the probability values by comparing it with the trained values. If the value is positive, negative, or more than 1, then this activation function converts them into 0 and 1

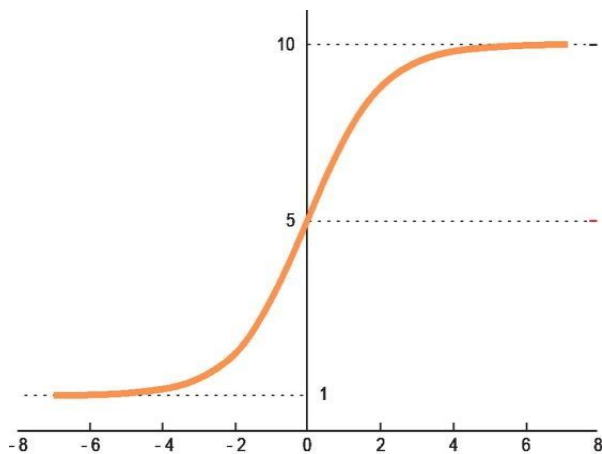


Fig 1.4: -Graph of SoftMax Activation Function

iv. OPTIMIZER

It functions as a technique for altering neural network properties like weight and learning rate. To achieve the weight's necessary ideal values during the training phase, it must be changed continuously. The Stochastic Gradient Descent optimizer is the most popular. We used the Adam, an SGD optimizer type, in our model. It uses a modified learning rate system to determine each variable's learning percentage. This contributes to decreasing overall loss and increases precision.

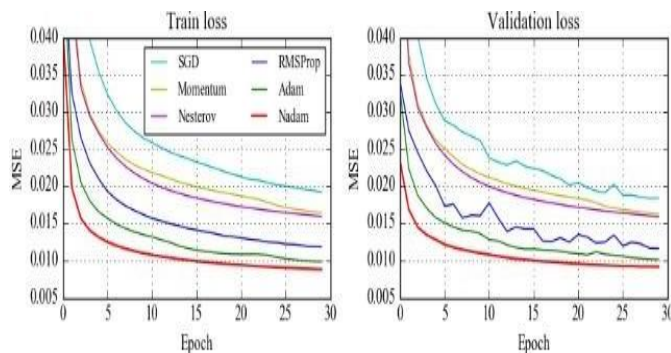


Fig 1.5: -Graph of the Adam Optimizer

III. OPEN CV

An image processing, machine learning, and computer vision library is called OpenCV. We utilize the Numpy library to manipulate read images into arrays so that we could process images and movies to recognize faces, objects, and other things.

Haar Cascade Algorithm:

This method, which is used for object recognition, can instantly recognise faces in an image or video. The boundary or line recognition features that viola and jones proposed in their paper "Rapid Item Detection Using a Boosted Cascade"

are used by this technique. Many positive and negative images are acclimated to learning in an ML-based manner.

IV. TENSOR FLOW

It is the second machine learning framework that Google has developed, and it offers a set of windows of operations for creating, designing, constructing, and training deep learning models in Python. Unlike the computers that use TensorFlow, Keras is a high-level API created for people.

OVERALL DESIGN OF THE PROPOSED SYSTEM:

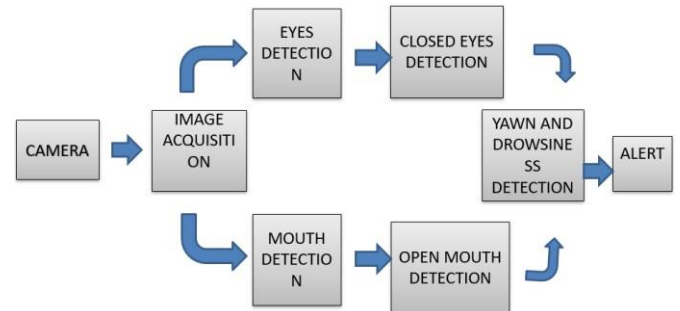
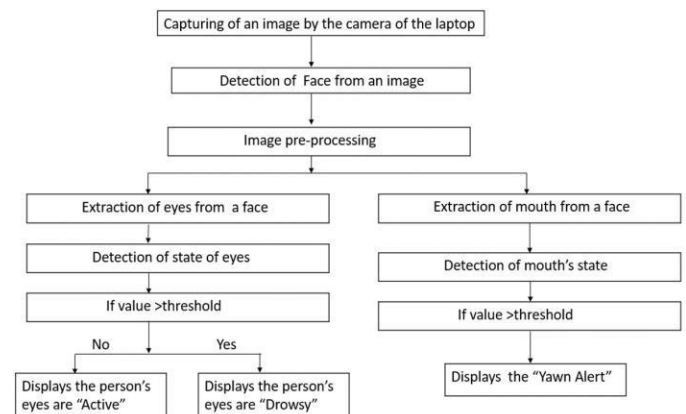


Fig2. System Architecture

The most frequent objection is that most of these systems merely used a pre-defined dataset of faces with closed eyes and open eyes to implement this problem statement. Additionally, they possessed only visual alerting systems to advise the driver of his condition, which are ineffective alarms because they necessitate alertness, which contradicts the very purpose of the warning.

STEPS INVOLVED IN THE ENTIRE PROJECT:



Also, it was discovered that some of these systems took too long to respond after determining the driver's state before warning him or her of it. Our solution seeks to resolve all of these problems with the pre-existing methods and do so while providing the greatest results accuracy possible. Our core concept is to use a live camera to observe the driver's physical condition as he or she is operating a vehicle. When a driver is fatigued and their eyes blink frequently, we use face parameters to monitor their retina in the beholder's eye as well as their mouth motions to detect yawning. Our device will immediately produce an alarm sound as loud as a siren alarm to immediately rouse the driver from his bad condition of back

to alertness when our system recognizes either of these changes.



a) Eyes



b) Mouth

V RESULTS AND DISCUSSION

A. Experiment setup

There are four experiments in this work: Face, eye, and mouth detection are the first three steps. 4) The eyelids being closed and the presence of yawning. Then, the author will place the infrared camera at the top of the dashboard, 15 degrees facing the driver, and the latter will exhibit sporadic behaviours suggestive of driver tiredness, such as frequent eye closure, frequent blinking, and sporadic yawning. These trials are conducted with four volunteers under low light. As seen in Figures 7a and 7b, the programme produces a green rectangle around the driver's eyes and lips when it recognises them. The programme prints a red rectangle at the symptom location, such as yawning, and a red rectangle surrounding the driver's mouth and eyes when it detects signs of driver drowsiness. Figures 7c demonstrate how the driver's eyes will be surrounded by a red rectangle on the closing.



a) Normal driver output



b) Normal driver output



c) Drowsy driver output

B. Accuracy Rate

There are four experiments, for example: 1) face detection 2) detection of eyes 3) detection of mouth 4) Detection of eye closure and yawning. Table I show examples.

Table-1. Accurate yawning and eyelashes closing detection

SNO.	Absolute Positive Rating (Eye)	Absolute Negative Rating (Eye)	Inaccurate Positive Rate (Mouth)	Inaccurate Negative Rate (Mouth)
1	32.2 %	96.3%	50.5 %	63.1 %
2	92.6 %	97.9%	96.5 %	96.2 %
3	75.4 %	95.4 %	77.1 %	69.5 %
4	99.2 %	98.6 %	96.9 %	97.0 %
Accuracy	97.0 %	96.5.0 %	91.7 %	94.5 %

VI. CONCLUSION

This paper presented an infrared camera-based method for detecting driver eye closure and yawning for drowsiness analysis. The four steps in this method are face detection, eye detection, mouth detection, and eyes closure and yawning detection. The effectiveness of the system was evaluated through four experiments: 1) Face detection, 2) Eyes detection, 3) Eyes closure and yawning detection, and 4) Mouth detection. The accuracy rates for identifying faces, eyes, mouths, and eyes closing and yawning were 99.47%, 94.33%, 99.80%, and 92.5%, respectively. When a face, like a hand, was obscured, mistakes happened. Although the camera's capture angle fluctuates in width, the technology should still be used successfully in subsequent research.

REFERENCES

- [1] Dr. K.S. Tiwari, Supriya Bhagat, Nikita Patil, Priya Nagare, IOT Based Driver Drowsiness Detection&Health Monitoring System.
- [2] Rateb Jabbar, Mohammed Shinoy, Mohamed Kharbeche, Khalifa Al-Khalifa, MoezKrichen, Kamel Barkaoui, Driver Drowsiness Detection Model using CNN Techniques for Android app.
- [3] Md. Yousuf Hossain, Fabian Parsia George, IOT based Real-time Drowsy Driving Detection System for the Prevention of Road Accidents.

- [4] Mika Sunagawa, Shin-ichi-Shikii, WataruNakai, Makoto Mochizuki, Koichi Kusakame, and Hiroki Kitajima, Comprehensive Drowsiness Level Detection Model Combining Multimodal Information.
- [5] Monagi H. Alkinani, Wazir Zada Khan, and Quratulain Arshad, Detecting Human Driver Inattentiveand Aggressive Driving Behavior using Deep Learning: Recent Advances, Requirements, and Open Challenges
- [6] MkhuseleNgxande, Jules-Raymond Tapamo and Michael Burke, Driver drowsiness detection using Behavioral measures and Machine Learning Techniques: A review of state-of-art techniques.
- [7] Joao Ruivo Paulo, Gabriel Pires, and Urbano J. Nunes, Cross-Subject Zero Calibration Driver's Drowsiness Detection: Exploring Spatiotemporal Image Encoding of EEG Signals for CNN Classification.
- [8] Naveen SenniappanKaruppusamy, Bo-Yeong Kang, Multimodal System to Detect Driver Fatigue using EEG, Gyroscope, and Image Processing.
- [9] Yaocong Hu, Mingqi Lu, Chao Xie, and Xiabo Lu, Driver Drowsiness Recognition via 3D Conditional GAN and Two-level Attention Bill STM.
- [10] CeerthiBala U.K, Sarath TV, Internet of things based Intelligent Drowsiness Alert System, Fifth International Conference on Communication and Electronics Systems