## ABSTRACT

With the world's population growing, distractions can cause dangerous accidents that are difficult to avoid. Several studies have been done in an effort to find a way to stop the loss of life. So, Accidents are more due to the driver’s drowsiness; it has been recorded that more than 40% of chances that accidents occur while the driver’s is in drowsiness state. It’s very important that the driver must be in alert state while driving the car. The protection of the passengers is the first concern of the vehicle's designers, and airbags are designed to improve security and save lives of occupants, but they do not stop accidents from occurring. The main reasons are general fatigue and distraction brought on by phone alerts. A wide range of techniques based on behavioral measures using machine learning techniques have been inspected to scope out driver distraction in the past. The recent growth of such technologies requires that these algorithms be improved to evaluate their accuracy in identifying distraction. There are numerous features of faces that are available to be extracted from any face to deduce the level of distraction. These include eyes closed for longer than 5 seconds, head movements and continuous yawning. However, the development of a shock system to push out the distraction and immediately give an alert is a challenging task as it requires accurate and robust algorithms. A novel system for evaluating the driver’s level of fatigue based on face tracking and facial key point detection. In order to track the driver’s face using CNN (Convolution Neural Network) and then the facial regions of detection based on facial key points. Then the eyes and mouth will be detected if the eye is closed the alert system will be displayed.

vi

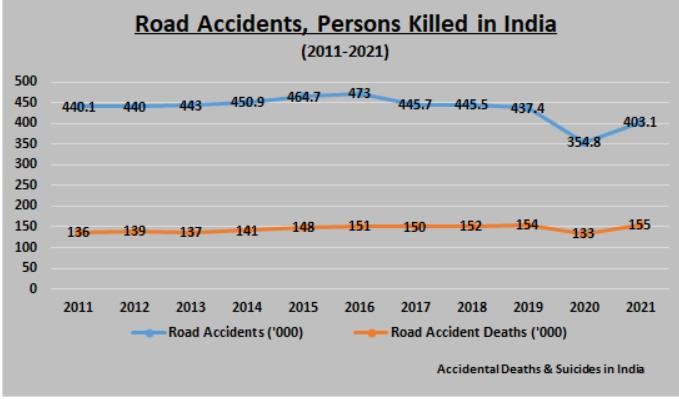
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **TABLE OF CONTENTS** | |  |
| **Chapter No.** |  | **TITLE** | | **Page No.** |
|  | **ABSTRACT** | | | v |
|  | **LIST OF FIGURES** | | | viii |
| 1 | **INTRODUCTION** | | | 1 |
| 2 | **LITERATURE SURVEY** | | | 4 |
|  | 2.1 | | **Inferences from Literature Survey** | 4 |
|  | 2.2 | | Open Problems in Existing System | 8 |
| 3 | **REQUIREMENTS ANALYSIS** | | | 9 |
|  | 3.1 | Feasibility Studies/Risk Analysis of the Project | | 9 |
|  | 3.2 | Software Requirements Specification Document | | 10 |
|  | 3.3 | System Use case | | 13 |
| 4 | **DESCRIPTION OF PROPOSED SYSTEM** | | | 16 |
|  | 4.1 | Selected Methodology or process model | | 16 |
|  | 4.2 | Architecture / Overall Design of Proposed System | | 23 |
|  | 4.3 | Description of Software for Implementation and Testing plan of the Proposed Model/System | | 24 |
|  | 4.4 | Project Management Plan | | 24 |
|  | 4.5 | Financial report on estimated costing | | 26 |
|  | 4.6 | Transition/ Software to Operations Plan | | 26 |
| **5** | **IMPLEMENTATION DETAILS** | | | 28 |
|  | 5.1 | Development and Deployment Setup | | 28 |
|  | 5.2 | Algorithms | | 29 |
|  | 5.3 | Testing | | 34 |
| **6** | **RESULTS AND DISCUSSION** | | | 39 |
| **7** | **CONCLUSION** | | | 40 |
|  | 7.1 | Conclusion | | 40 |
|  | 7.2 | Future work | | 40 |
|  | 7.3 | Research Issues | | 41 |
|  | 7.4 | Implementation Issues | | 41 |
|  | **REFERENCES** | | | 42 |

|  |  |
| --- | --- |
| **APPENDIX** | 44 |
| **A. SOURCE CODE** | 44 |
| **B. SCREENSHOTS** | 52 |
| **C. RESEARCH PAPER** | 54-59 |

vii

|  |  |  |
| --- | --- | --- |
|  | **LIST OF FIGURES** |  |
| **FIGURE**  **NO** | **FIGURE NAME** | **Page**  **No.** |
| 1.1 | Road Accidents,Persons Killed in India | 1 |
| 1.2 | State-Wise Road Accidents Reported | 2 |
| 1.3 | Detection | 3 |
| 3.3.1.1 | Use Case Diagram | 13 |
| 3.3.2.1 | Activity Diagram | 14 |
| 3.3.3.1 | Collaboration Diagram | 15 |
| 3.3.4.1 | Sequence Diagram | 15 |
| 4.1.1 | Drowsy by EAR | 23 |
| 4.2.1 | Architecture Diagram | 23 |
| 5.2.3.1 | Outflow of CNN | 31 |
| 5.2.3.2 | A Full CNN | 33 |
| B.1 | Output of enhanced Eye CNN | 52 |
| B.2 | Face Recognition with specs | 52 |
| B.3 | Face Recognition without specs | 52 |
| B.4 | Detection when Eyes are open | 52 |
| B.5 | Detection when Eyes are closed | 52 |
| B.6 | Eye Detection with specs | 53 |
| B.7 | When a Chauffeur is non-Torpid | 53 |
| B.8 | Yawn Detection | 53 |
| B.9 | Torpidity Detection | 53 |

# CHAPTER 1 INTRODUCTION

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. Improvement of technologies for recognizing or avoiding laziness at the wheel could be a major challenge within the field of accident evasion frameworks. Due to the risk that laziness or fatigue presents on the street, strategies have to be shaped for neutralizing its influences. Both driver tiredness and diversion, in any case, might have the same impacts, i.e., diminished driving execution, longer response time, and an expanded hazard of crash inclusion. Based on Procurement of video from the camera that's before driver, it performs real-time preparing of an approaching video stream in order together the driver‘s level of weariness on the off chance that the laziness is estimated at that point it'll deliver the caution by detecting the eyes, mouth and head posture.

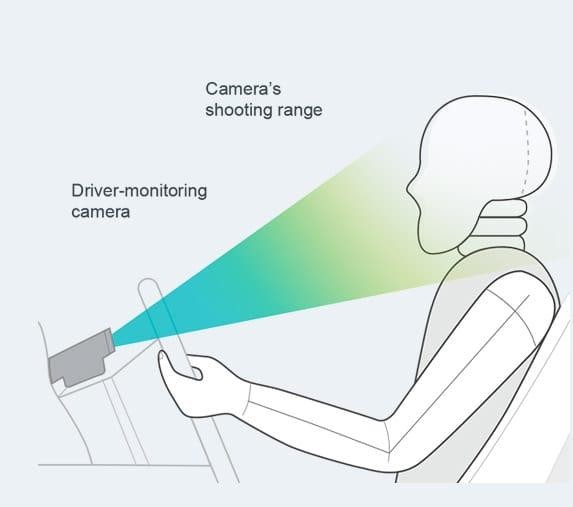
**Figure.1. 1:Road Accidents,Persons Killed in India**

In recent years, ansurge in the demand for modern transportation simultaneously demands a faster car safety growth. At present, the automobile is the most essential mode of transportation for people. Although it has changed people‘s

lifestyle and improved the convenience of conducting daily activities, it is also associated with numerous negative side-effects, such as road mishaps and traffic due to distraction, fatigue, and exhaustion. 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 characterizedas 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.

**Figure.1.2:State-Wise Road Accidents Reported**

This project proposed that yawning and eyes detection is the obvious signs of fatigue and drowsiness.Driver Drowsiness Detection by Using Webcam is being introduced to minimize and reduce the number of accidents involving cars, lorries and trucks. It detects the drowsiness signs and alerts drivers when they are in drowsy state.



## Scope of Study

**Figure.1. 3 Detection**

In this project, the author will focus on these following procedures:

* Basic concept of drowsiness detection system
* Familiarize with the signs of drowsiness
* Determine the drowsiness from these parameters - Eye blink - Area of the pupils detected at eyes - Yawning
* Data collection and measurement.
* Integration of the methods chosen.
* Coding development and testing.
* Complete testing and improvement

## Relevancy of the Project

This project is relevant to the implementation since fatigue and drowsiness drivers contribute to the percentage of road accidents. Many researches have been conducted to implement safe driving systems in order to reduce road accidents. Detecting the driver’s alertness and drowsiness is an efficient way to prevent road accidents. With this system, drivers who are drowsy will be alerted by an alarm to regulate consciousness, attention and concentration of the drivers. This will help to reduce the number of road accidents

# CHAPTER 2 LITERATURE SURVEY

This problem statement has been extensively studied over the past 3 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.

1. **Title:** EEG-based Cross-Subject Driver Drowsiness Recognition with an Interpretable CNN [Year:2022]

**Author:** Jian Cui, Olga Souria, Wolfgang Müller-Wittig, zirui Lan

## Description:

In the context of electroencephalogram (EEG)-based driver drowsiness recognition, it is still challenging to design a calibration-free system, since EEG signals vary significantly among different subjects and recording sessions. Many efforts have been made to use deep learning methods for mental state recognition from EEG signals. However, existing work mostly treats deep learning models as black-box classifiers, while what have been learned by the models and to which extent they are affected by the noise in EEG data are still underexplored. In this paper, we develop a novel convolutional neural network combined with an interpretation technique that allows sample-wise analysis of important features for classification. Then, edge detection is used to locate the regions of eyes. In addition to being used as the &namic templates for eye tracking in the next frame, the obtained eyes’ images are also used for fatigue detection in order to generate some warning alarms for driving safety The network has a compact structure and takes advantage of separable convolutions to process the EEG signals in a spatial-temporal sequence. Results show that the model achieves an average accuracy of 78.35% on 11 subjects for leave-one-out cross-subject drowsiness recognition, which is higher than the conventional baseline methods of 53.40%-72.68% and state-of-the-art deep learning methods of 71.75%-75.19%. Interpretation results indicate the model has learned to recognize biologically meaningful features from EEG signals, e.g., Alpha spindles, as strong indicators of drowsiness across different subjects.

1. **Title:** Driver’s Fatigue Detection Based on Yawning Extraction [Year:2022]

**Author:** Aouatif Amine,1,2 ,Mohammed Rziza1, Nawal Alioua.

## Description:

The increasing number of traffic accidents is principally caused by fatigue. In fact, the fatigue presents a real danger on road since it reduces driver capacity to react and analyze information. In this paper we propose an efficient and nonintrusive system for monitoring driver fatigue using yawning extraction. The proposed scheme uses face extraction based support vector machine (SVM) and a new approach for mouth detection, based on circular Hough transform (CHT), applied on mouth extracted regions. Our system does not require any training data at any step or special cameras. Some experimental results showing system performance are reported.These experiments are applied over real video sequences acquired by low cost web camera and recorded in various lighting conditions.

1. **Title:** Driver’s Drowsiness Detection Based on Behavioural Changes using ResNet [Year:2021]

**Author:** A. Jeyasekar, Vivek Ravi Iyengar

## Description:

Recently there has been growing interest in intelligent transportation system because the road accidents become biggest problems of mankind and the casualties of accident also increases rapidly every year. The casualties are very often witnessed in heavy and light motor vehicles. Moreover, the accidents occur mainly due to carelessness and drowsy feeling of the driver. Intelligent transportation systems use deep learning mechanism to detect drowsiness of the driver and alert the same to driver. It results in reduction of accidents. The driver’s behaviour during drowsiness is detected by three types of approaches. One approach deploys the sensors in steering wheel and accelerator of the vehicle and analyzes the signal sent by the sensors to detect the drowsiness. Second approach focuses on measuring the heart rate, pulse rate and brain signals etc to predict the drowsiness. Third approach uses the facial expression of the driver such as blinking rate of eye, eye closure and yawning etc. The cause for most of the road accidents is driver’s drowsiness. Therefore, in this paper, the behavioural changes of driver is accounted to detect the drowsiness of the driver. Eye movement and yawning are two behavioural changes of driver is considered in this paper.

1. **Title:** Driver Fatigue Detection Based on Eye Tracking [Year:2021] **Author:** Chuu-Hai Fan, Chih-Yuan Chen, Wen-Bing Horng ,Yi Chang **Description:**

A vision-based re al-time driver fatigue detection system is proposed for driving safely. The driver’s face is located, from color images captured in a car, by using the characteristic of skin colors. Then, edge detection is used to locate the regions of eyes. In addition to being used as the &namic templates for eye tracking in the next frame, the obtained eyes’ images are also used for fatigue detection in order to generate some warning alarms for driving safety. The system is tested on a Pentium Ill 550 CPU with 128 MB RAM. The experiment results seem quite encouraging and promising. The system can reach 10 fiames per second for eye tracking, and the average correct rate for eye location and tracking can achieve 79.1% on four test videos. The correct rate for fatigue detection is 8o%, but the average precision rate is 78.9% on the test videos.

1. **Title:** Facial Expression Recognition using Convolutional Neural Network with Data Augmentation [Year:2021]

**Author:** Tawsin Uddin Ahmed, Sazzad Hossain, Mohammad Shahadat Hossain, Raihan Ul Islam, Karl Andersson

## Description:

Detecting emotion from facial expression has become an urgent need because of its immense applications in artificial intelligence such as human-computer collaboration, data-driven animation, human-robot communication etc. Since it is a demanding and interesting problem in computer vision, several works had been conducted regarding this topic. Then, edge detection is used to locate the regions of eyes. In addition to being used as the &namic templates for eye tracking in the next frame, the obtained eyes’ images are also used for fatigue detection in order to generate some warning alarms for driving safety Our system does not require any training data at any step or special cameras. The objective of this research is to develop a facial expression recognition system based on convolutional neural network with data augmentation. This approach enables to classify seven basic emotions consist of angry, disgust, fear, happy, neutral, sad and surprise from image data. Convolutional neural network with data augmentation leads to higher validation accuracy than the other existing models (which is 66.24%) as well as helps to overcome their limitations.

1. **Title:** Deep Neural Network for Human Face Recognition [Year:2021] **Author:** Dr. Priya Guptaa, Jagriti Tripathia, Nidhi Saxenaa, Meetika Sharmaa, **Description:**

Face recognition (FR), the process of identifying people through facial images, has numerous practical applications in the area of biometrics, information security, access control, law enforcement, smart cards and surveillance system. Convolutional Neural Networks (CovNets), a type of deep networks has been proved to be successful for FR. For real-time systems, some preprocessing steps like sampling needs to be done before using to CovNets. But then also complete images (all the pixel values) are passed as input to CovNets and all the steps (feature selection, feature extraction, training) are performed by the network. This is the reason that implementing CovNets are sometimes complex and time consuming. CovNets are at the nascent stage and the accuracies obtained are very high, so they have a long way to go. The paper proposes a new way of using a deep neural network (another type of deep network) for face recognition. In this approach, instead of providing raw pixel values as input, only the extracted facial features are provided. This lowers the complexity of while providing the accuracy of 77.05% on Yale faces dataset.

1. **Title:** Driver Fatigue Detection Based on Facial Key Points and LSTM[Year:2020]

**Author:** Long Chen,1 Guojiang Xin ,2 Yuling Liu,1 and Junwei Huang3

## Description:

Recent years, fatigue driving has been a serious threat to the traffic safety, which makes the research of fatigue detection a hotspot field. Research on fatigue recognition has a great significance to improve the traffic safety.. In order to detect whether the driver has fatigue driving, this paper proposes a fatigue state recognition algorithm ,the method first uses MTCNN (multitask convolutional neural network) to detect human face, and then DLIB (an open-source software library) is used to locate facial key points to extract the fatigue feature vector of each frame. The fatigue feature vectors of multiple frames are spliced into a temporal feature sequence and sent to the LSTM (long short-term memory) network to obtain a final fatigue feature value. Experiments show that compared with other methods, the fatigue state recognition algorithm proposed in this paper has achieved better results in accuracy. The average accuracy of the proposed method in detecting key points of the face is as high as 73%, and the running time is less than half of the ordinary DLIB method.

## INFERENCES FROM LITREATURE SURVEY

In recent years, many scholars and institutions have conducted a lot of researches on the fatigue driving detection based on the computer vision. In Existing study, a low cost, real time driver’s drowsiness detection systems is developed with some acceptable accuracy. Facial landmarks on the detected face are pointed and subsequently the eye aspect ratio and mouth opening ratio are computed and depending on their values.

Features of the eyes and mouth were used to model the behavior of the Chauffeur. Changes in these characteristics were used to monitor the Yawning and frequent blinking are the most obvious signs of driver fatigue. Therefore, first task is to determine the human eyes’ state and mouth’s state. there are generally two ways to detect the eyes and mouth. One is to directly detect the positions of the eyes and mouth. Other is to firstly find the facial area and then detect the positions of the eyes and mouth. Human face has more information and the features are more stable than the human eyes. Cutting out the face area can reduce the test range of the eye position and avoid the interference of the background.

## OPEN PROBLEMS IN EXISTING SYSTEM

The driver facial features are identified by EEG. But it is not accurate and some major problems are occurred. The Detect faces are pointed and this point gives eye aspect ratio. But the ratio is not working properly.

Therefore, the quality of system input determines the quality of system output. Well-designed input forms and screens have following properties −

* + - It should serve specific purpose effectively such as storing, recording, and retrieving the information.
    - Ensures proper completion with accuracy.
    - Requires more ram and memory to process.
    - Easy to fill and straightforward.
    - High Running time.
    - It does not generate alarm.

# CHAPTER 3 REQUIREMENT ANALYSIS

## FEASIBILITY STUDIES/RISK ANALYSIS OF THE PROJECT FEASIBILITY STUDY

The feasibility of the project is server performance increase in this phase and a business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis, the feasibility study of the proposed system is to be carried out. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* Economical feasibility
* Technical feasibility
* Operational feasibility

## ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of funds that the company can pour into the research and development of the system is limited. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

## TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands being placed on the client. The developed system must have modest requirements, as only minimal or null changes are required for implementing this system.

## OPERATIONAL FEASIBILITY

The aspect of the study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently.. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

## SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

A software requirements specification (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide. In order to fully understand one’s project, it is very important that they come up with a SRS listing out their requirements, how are they going to meet it and how will they complete the project. It helps the team to save upon their time as they are able to comprehend how are going to go about the project. Doing this also enables the team to find out about the limitations and risks early on. Requirement is a condition or capability to which the system must conform. Requirement Management is a systematic approach towards eliciting, organizing and documenting the requirements of the system clearly along with the applicable attributes. The elusive difficulties of requirements are not always obvious and can come from any number of sources.

Requirement’s analysis is very critical process that enables the success of a system or software project to be assessed. Requirements are generally split into two types: Functional and non-functional requirements. Functional Requirements: These are the requirements that the end user specifically demands as basic facilities that the system should offer.

## Functional Requirement

Functional Requirement defines a function of a software system and how the system must behave when presented with specific inputs or conditions. These may include calculations, data manipulation and processing and other specific functionality. Following are the functional requirements on the system:

1. All the data must be in the same format as a structured data.
2. The data collected will be vectorized and sent across to the classifier.

## Examples of functional requirements:

1. Authentication of user whenever he/she logs into the system
2. System shutdown in case of a cyber-attack
3. A verification email is sent to user whenever he/she register for the first time on some software system.

## Non-Functional Requirements

Non-functional requirements are the requirements which are not directly concerned with the specific function delivered by the system. They specify the criteria that can be used to judge the operation of a system rather than specific behaviors. They may relate to emergent system properties such as reliability, response time and store occupancy. Non-functional requirements arise through the user needs, because of budget constraints, organizational policies and the need for interoperability with other software and hardware system.

These are basically the quality

constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements:

Portability Security Maintainability Reliability Scalability Performance Reusability

## Examples of non-functional requirements:

1. Emails should be sent with a latency of no greater than 12 hours from such an activity.
2. The processing of each request should be done within 10 seconds
3. The site should load in 3 seconds whenever of simultaneous users are &gt; 10000

## SYSTEM CONFIGURATION Software Requirements:

* + - * Operating System : Windows 7, 8, 10 (64 bit)
      * Software : Python
      * Tools : Python 3.7 IDLE , Anaconda , Jupyter
      * Webcam : 1

## Hardware Requirements:

* Hard Disk : 500GB and Above
* RAM : 4GB and Above
* Processor : I3 and Above
* Webcam : 1
* Speaker : 1

## TECHNOLOGIES USED

* + Python

# Introduction to Python

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

It is used for:

* web development (server-side),
* software development,
* mathematics,
* System scripting.

## SYSTEM USE CASE UML DIAGRAMS

* + 1. **Use Case Diagram:**

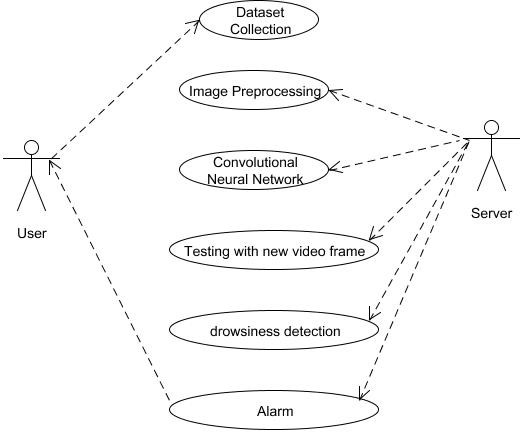
Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of software engineering. The standard is managed and was created by the Object Management Group. UML includes a set of graphic notation techniques to create visual models of software intensive systems. This language is used to specify, visualize, modify, construct and document the artifacts of an object oriented software intensive system under development.

A Use case Diagram is used to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases.

Use case diagram consists of two parts:

**Use case:** A use case describes a sequence of actions that provided something of measurable value to an actor and is drawn as a horizontal ellipse.

**Actor:** An actor is a person, organization or external system that plays a role in one or more interaction with the system.



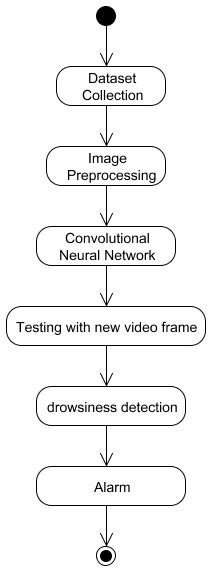
**fig 3.3.1.1 Use Case Diagram**

## Activity Diagram:

Activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram shows the overall flow of control.

The most important shape types:

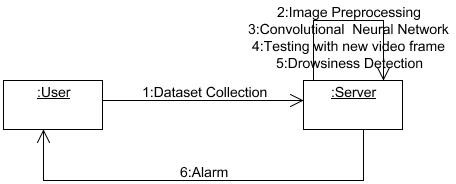
* Rounded rectangles represent activities.
* Diamonds represent decisions.
* Bars represent the start or end of concurrent activities.
* A black circle represents the start of the workflow.
* An encircled circle represents the end of the workflow.



**fig 3.3.2.1 Activity Diagram**

## Collaboration Diagram:

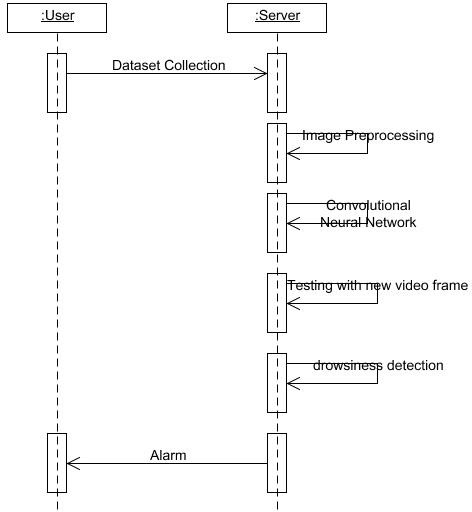
UML Collaboration Diagrams illustrate the relationship and interaction between software objects. They require use cases, system operation contracts and domain model to already exist. The collaboration diagram illustrates messages being sent between classes and objects.



## Sequence Diagram:

**3.3.3.1 Collaboration Diagram**

A Sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of Message Sequence diagrams are sometimes called event diagrams, event sceneries and timing diagram.



**3.3.4.1 Sequence Diagram**

# CHAPTER 4 DESCRIPTION OF PROPOSED SYSTEM

## SELECTED METHODOLOGY OR PROCESS MODEL 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. Other systems gave alarms continuously for a long time resulting in spamming the system. system aims to overcome all these issues with the previously existing systems and address these issues while giving the best accuracy in the results. Basic idea is to monitor the physical state of the driver while he is driving using a live camera. 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 system detects either of these changes, model will immediately emit an alarm sound as loud as a siren alarm to immediately awaken the driver from his poor state back.

## Problem Statement

Current drowsiness detection systems monitoring the driver’s condition requires complex computation and expensive equipment, not comfortable to wear during driving and is not suitable for driving conditions; for example, Electroencephalography (EEG) and Electrocardiography (ECG), i.e. detecting the brain frequency and measuring the rhythm of heart, respectively. A drowsiness detection system which use a camera placed in front of the driver is more suitable to be use but the physical signs that will indicate drowsiness need to be located first in order to come up with a drowsiness detection algorithm that is reliable and accurate. Lighting intensity and

while the driver tilt their face left or right are the problems occur during detection of eyes and mouth region. Therefore, this project aims to analyze all the previous research and method, hence propose a method to detect drowsiness by using video or webcam. It analyzes the video images that have been recorded and come up with a system that can analyze each frame of the video.

## Libraries Used

* + - OpenCV: Main objective of this library is to be able to provide an infrastructure for real-time computer vision. In this prototype, it will be used to access the camera and
    - Haar feature-based classifiers for object detection.
    - TensorFlow: Used for implementation of large-scale machine learning algorithms and numerical computation. Also, a prerequisite for the implementation of Keras.
    - NumPy: Used for creation and working of arrays in python for multiple aspects.
    - Imutils: Majorly used for pre-processing of images like resizing, translation, rotation and skeletonization.
    - Keras: A neural network library used for developing classification models.

## DLib Library

DLib is an open-source C++ library which houses implementations of algorithms for performing inference in Bayesian networks and kernel-based methods for classification, regression, clustering, anomaly detection, and feature ranking.

DLib also features utility functionality including Threading, Networking, Numerical Algorithms, Image Processing, Data Compression and Integrity algorithms

## MODULES

* Dataset Collection
* Convolutional Neural Network
* Detection

## MODULE EXPLANATION:

* + 1. **Dataset Collection:**

Our Real-Time Driver-Drowsiness Detection project dataset are collected from kaggle.com. When the images are collected, pre-process the image after collection of various records. In this preprocessing step, images are made ready, for the algorithm.

## Convolutional Neural Network:

The angle of the eye to compensate for the CNN's limitations regarding eye closure recognition. After the CNN validates that the driver's eye is open, we use the angle of the eye to validate the result. A blink is the process of the eye closing and opening.

## Detection:

When the program is initiated, the camera will turn on. And the camera will start recording continuously. In this way, it monitors the driver every second. Once the driving is falling into sleep the alarm start notification to wake-up the driver from sleep.

## Loading the dataset

The dataset is conveniently provided to us as part of the Keras library, so we can easily load the dataset.

When we load the dataset below, X\_train and X\_test will contain the images, and y\_train and y\_test will contain the digits that those images represent.

Exploratory data analysis

Now let’s take a look at one of the images in our dataset to see what we are working with. We will plot the first image in our dataset and check its size using the ‘shape’ function.

By default, the shape of every image in the mnist dataset is 28 x 28, so we will not need to check the shape of all the images. When using real-world datasets, you may not be so lucky. 28 x 28 is also a fairly small size, so the CNN will be able to run over each image pretty quickly.

## Data pre-processing

Next, need to reshape our dataset inputs (X\_train and X\_test) to the shape that our model expects when we train the model. The first number is the number of images (60,000 for X\_train and 10,000 for X\_test). Then comes the shape of each image (28x28). The last number is 1, which signifies that the images are greyscale.

The need to ‘one-hot-encode’ our target variable. This means that a column will be created for each output category and a binary variable is inputted for each category. For example,it take first image in the dataset is a 5. This means that the sixth number in our array will have a 1 and the rest of the array will be filled with 0.

## Working Procedure:

Model is built using Convolutional Neural Networks (CNN).

A convolutional neural network is a special type of deep neural network which performs extremely well for image classification purposes.A CNN basically consists of an input layer, an output layer and a hidden layer which can have multiple numbers of layers.

**Step 1** - Take Image as Input from a Camera With a webcam.

It take images as input. So to access the webcam, and made an infinite loop that captures each frame.

**Step 2** - Detect Face in the Image and Create a Region of Interest(ROI).

To detect the face in the image, then need to first convert the image into grayscale as OpenCV algorithm for object detection takes gray images in the input.’Don't need color information to detect the objects.

**Step 3 -** Detect the eyes from ROI and feed it to the classifier(CNN). The same procedure to detect faces is used to detect eyes.

## Facial landmark detection:

Individual frames are taken from the input video stream and are converted from RGB to grayscale using openCV. At that point, individual frames are extracted and are changed over from RGB to grayscale utilizing openCV. From that point forward, dlib is utilized to identify the face and plot the tourist spots. This is finished utilizing the dlib shape indicator record which has the identifier and the indicator. The indicator is utilized to distinguish the face and the indicator is utilized to plan the facial tourist spots, for example, the eyes, mouth, nose, eyebrows, jaw from the edge. At that point, a facial milestone finder which is a piece of the dlib library is utilized to plot 68 focuses on the face which guide to different facial structures like the eyes, lips, nose, eyebrows, and the jaw as appeared. We at that point use face utils from the library imutils to remove files of the necessary facial milestones, The tourist spots of enthusiasm for the instance of this venture are the eyes and the mouth district. Utilizing these lists or facial focuses, we can ascertain the eye angle ratio (EAR) and the lip separation.

## Eye blink detection:

The face and eye identification are found through opencv and dlib. We can apply facial milestone location to confine significant locales of the face, including eyes, eyebrows, nose, ears, and mouth. This methodology is used for measuring the state of the eye. As far as flicker identification, we are just keen on two arrangements of facial structures - the eyes.After localizing the eye region the state of the eye is measured i.e. if the eye is closed or opened. Eye aspect ratio (EAR) is used to measure the state of the eye.

The full set of facial landmarks that can be detected using dlib each eye is represented by 6 (x, y)- coordinates, starting at the left corner of the eye (as if you were looking at the person), and then working clockwise around the remainder of the region. Based on this image, we should take away one key point. There is a relation between the width and the height of these coordinates. We can then derive an equation that reflects this relation called the eye aspect ratio (EAR):

EAR = ǁǁ p2– p6 ǁǁ + ǁǁ p3 – p5 ǁǁ / 2 ǁǁ p1 – p4 ǁǁ (1)

Where p1... p6 are 2D facial landmark locations.The numerator of this equation computes the distance between the vertical eye landmarks. The denominator computes the distance between horizontal eye landmarks, weighting the denominator appropriately since there is only one set of horizontal points but two sets of vertical points.If the EAR value is high, it means the numerator is higher than the denominator, which says the average of the vertical distance is greater than the horizontal distance and hence the eyes are opened. 2a. If the EAR value is approaching zero, it means the numerator is lesser than the denominator, which portrays that the average of the vertical distance is lesser than the horizontal distance and hence the eyes are closed. 2b. This methodology is used for measuring the state of the eye.

## Yawn detection:

He mouth region is detected through dlib. Facial landmark detection is applied to localize important regions of the face.For yawn detection, at first, the top lip is localized using a full set of facial landmark detection via dlib and likewise the lower lip. The mean of the top lip is calculated and the same is calculated for the lower lip. The absolute distance is calculated from the top lip to the lower lip.

## Software implementation:

At the point when the framework is begun, the counter, the libraries, and the camera are instated. On the off chance that everything is instated effectively, we go ahead and the camera begins catching. At that point, singular edges are separated and are changed over from RGB to grayscale utilizing openCV. From that point forward, dlib is utilized to distinguish the face and plot the tourist spots. This is finished utilizing the dlib shape indicator record which has the locator and the indicator. The identifier is utilized to identify the face and the indicator is utilized to plan the facial tourist spots, for example, the eyes, mouth, nose, eyebrows, and jaw from the casing.

Once we calculate EAR, we compare it with the threshold EAR value, which is

0.3. If the EAR value is less than the threshold EAR value, the eyes are considered shut and the counter is incremented. Then, EAR value of subsequent frames is also lesser than the threshold value, the counter is again incremented. The counter value is incremented until a frame has EAR value greater than the threshold value, which then leads to the resetting of the counter. If the counter value is above the threshold number of frames, which is 50, Similarly, we take the facial points or the indexes of the mouth region to find the lip distance, which translates to the distance between the lips. Once the lip distance is calculated for a frame, it is compared with its threshold value. If the calculated lip distance is greater than the threshold lip distance, which is 30, the driver is given a yawn alert.

**Step 4 -** Classifier Categorizes whether Eyes are Open or Closed. CNN classifier used for predicting the eye status. To feed image into the model. First, convert color image into grayscale. Then, resize the image to 24\*24 pixels as model was trained on 24\*24 pixel images, Normalize data for better convergence Now we predict each eye with model then, it states that eyes are closed.

## Building the model

Now we are ready to build our model. Here is the code: The model type that we will be using is Sequential. Sequential is the easiest way to build a model in Keras. It allows you to build a model layer by layer. We use the ‘add()’ function to add layers to our model. Our first 2 layers are Conv2D layers. These are convolution layers that will deal with our input images, which are seen as 2-dimensional matrices.64 in the first layer and 32 in the second layer are the number of nodes in each layer. This number can be adjusted to be higher or lower, depending on the size of the dataset. In our

case, 64 and 32 work well, so we will stick with this for now. Kernel size is the size of the filter matrix for our convolution. So a kernel size of 3 means we will have a 3x3 filter matrix. Refer back to the introduction and the first image for a refresher on this.

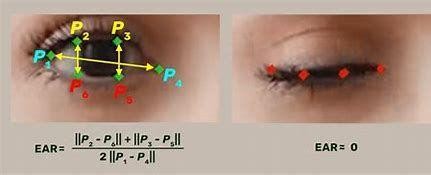
Activation is the activation function for the layer. The activation function we will be using for our first 2 layers is the ReLU, or Rectified Linear Activation. This activation function has been proven to work well in neural networks.Our first layer also takes in an input shape. This is the shape of each input image, 28,28,1 as seen earlier on, with the 1 signifying that the images are greyscale. In between the Conv2D layers and the dense layer, there is a ‘Flatten’ layer. Flatten serves as a connection between the convolution and dense layers. Dense is the layer type we will use in for our output layer. Dense is a standard layer type that is used in many cases for neural networks.We will have 10 nodes in our output layer, one for each possible outcome (0–9).The activation is ‘softmax’. Softmax makes the output sum up to 1 so the output can be interpreted as probabilities.

The optimizer controls the learning rate. We will be using ‘adam’ as our optmizer. Adam is generally a good optimizer to use for many cases. The adam optimizer adjusts the learning rate throughout training. The learning rate determines how fast the optimal weights for the model are calculated. We will use ‘categorical\_crossentropy’ for our loss function.

## Using our model to make predictions:

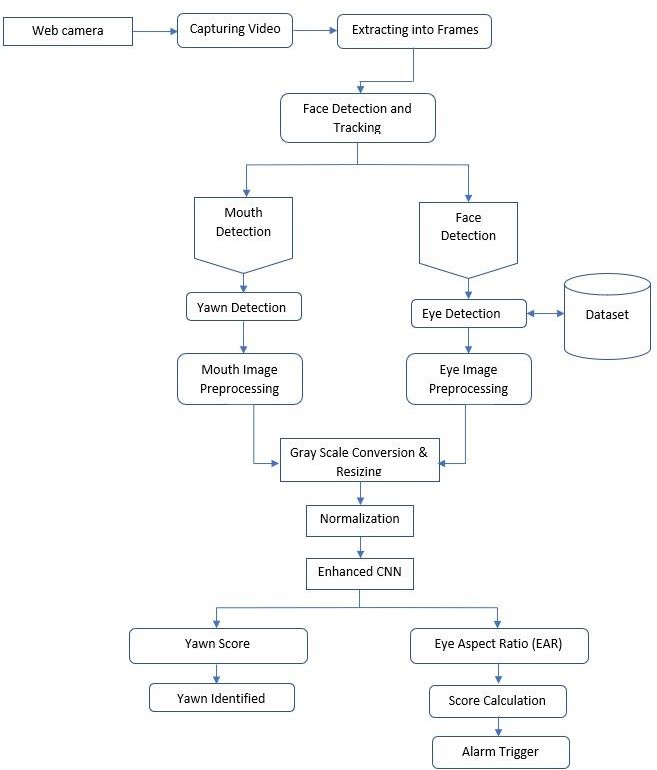
If you want to see the actual predictions that our model has made for the test data, we can use the predict function. The predict function will give an array with 10 numbers. These numbers are the probabilities that the input image represents each digit (0–9). The array index with the highest number represents the model prediction. The sum of each array equals 1 (since each number is a probability).

**Step 5 -** Calculate Score to Check whether Person is Drowsy,as fig: The score is basically a value to determine how long the person has closed his eyes. To feed image into the model. First, convert color image into grayscale. Then, resize the image to 24\*24 pixels as model was trained on 24\*24 pixel images, Normalize data for better convergence Now we predict each eye with model then, it states that eyes are closed.So, if both eyes are closed, then it keeps on increasing score and when eyes are open, then score decrease.



**Figure 4.1.1 Drowsy by EAR**

## ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM



**Figure 4.2.1 Architecture Diagram**

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

The Driver Drowsiness Detection system will be developed using Python programming language and the TensorFlow library for training and testing the CNN model. The system will also use OpenCV library for real-time video processing.

## The software will consist of the following modules:

**Data Collection and Preprocessing Module:** This module will be responsible for collecting facial expression and movement data from a video camera and preprocessing it for use in training the CNN model.

**CNN Model Training and Testing Module:** This module will train the CNN model using the preprocessed data and evaluate the model's performance using testing data.

**Real-time Video Processing Module**: This module will capture video data from a camera and process it in real-time to detect driver drowsiness. The module will use the trained CNN model to analyze facial expressions and movements to determine if the driver is becoming drowsy.

**Alarm/Alert Module:** This module will be responsible for triggering an alarm or alert to notify the driver when drowsiness is detected.

**User Interface Module:** This module will provide a user interface for the system, allowing the user to configure system settings, view system status, and receive alerts.

# PROJECT MANAGEMENT PLAN

**Project Title:** Chauffeur Torpidity Forewarning Aid using CNN

**Project Goal:** The goal of this project is to develop a system that can detect driver drowsiness using a Convolutional Neural Network (CNN) algorithm. The system will use a camera to monitor the driver's face, and the CNN model will analyze the facial expressions and movements to determine if the driver is becoming drowsy or sleepy. If the system detects drowsiness, it will alert the driver to take a break or stop driving.

## Project Phases:

1. **Planning Phase:**
   * Define the project goal and objectives
   * Identify project stakeholders and their roles and responsibilities
   * Establish project timelines and milestones
   * Determine project budget and resource requirements
   * Create a project charter and obtain approval from stakeholders

## Design Phase:

* + Research and select appropriate CNN model for driver drowsiness detection
  + Collect and preprocess facial expression and movement data for training the CNN model
  + Design and develop the software architecture for the driver drowsiness detection system
  + Create a user interface for the system

## Development Phase:

* + Train the CNN model using the preprocessed data
  + Integrate the CNN model into the software architecture
  + Develop a real-time video processing module to analyze the driver's face in real-time
  + Implement an alarm/alert system to notify the driver when drowsiness is detected

## Testing Phase:

* + Conduct unit testing on individual modules of the system
  + Conduct integration testing to ensure all modules are functioning correctly
  + Conduct system testing to ensure the system is functioning as expected
  + Conduct user acceptance testing with a group of drivers to ensure the system is user-friendly and effective

## Deployment Phase:

* + Deploy the driver drowsiness detection system to a real-world environment
  + Provide training and support to users of the system
  + Conduct ongoing monitoring and maintenance of the system

## Project Management Approach:

The project will follow an agile project management approach. The project team will work in sprints of two weeks each, with a sprint review and planning meeting held at the end of each sprint. The project manager will hold regular meetings with the project team to review progress, address issues and risks, and ensure that the project is on track. The project team will use a project management tool, such as JIRA, to track tasks, issues, and milestones.

**PROJECT MANAGEMENT PLAN**

|  |  |
| --- | --- |
| **Introduction** | **September 1-30** |
| **Literature Survey** | **October 1-31** |
| **System Design** | **November 1-30** |
| **System Implementation** | **December 1-31** |
| **Testing** | **January 1-30** |

Fig No.4.4.1 Project Management Plan

## Financial report on estimated costing:

It is based on quality of implementation the cost changes present we proved in system so in system its free of cost in implementation the cost will change based on quality and implementation.

## TRANSITION / SOFTWARE TO OPERATIONS PLAN Transition to Operations Plan:

After the Driver Drowsiness Detection system has been developed, tested, and approved for deployment, the project will transition to an operations phase. During this phase, the focus will be on deploying the system to a real-world environment and ensuring that it is functioning correctly.

The following steps will be taken to ensure a smooth transition to operations:

**Deployment Plan:** A deployment plan will be created to ensure that the system is deployed efficiently and effectively. The plan will include instructions for installing and configuring the software, hardware, and necessary components.

**User Training:** A training program will be developed to educate users on how to use the system. The program will include information on how to configure the system, interpret system results, and respond to system alerts.

**Support Plan**: A support plan will be created to ensure that users have access to support in case of any issues or questions. The plan will include contact information for support personnel and instructions on how to report issues.

**Maintenance Plan:** A maintenance plan will be created to ensure that the system is maintained properly. The plan will include instructions for regular system maintenance tasks, such as updating the software and hardware components, and ensuring that the system is functioning correctly.

**Performance Monitoring:** The system will be monitored regularly to ensure that it is functioning correctly and providing accurate results. The monitoring will include the analysis of system logs, performance metrics, and user feedback.

**Continuous Improvement:** The system will be continuously improved based on user feedback, system performance metrics, and emerging technologies. The project team will work with users to identify areas for improvement and implement changes as necessary.

The project team will work closely with users to ensure that the transition to operations is smooth and that the system is functioning correctly. The project manager will monitor the transition to operations phase closely to ensure that the system is meeting the project goals and objectives.

# CHAPTER-5 IMPLEMENTATION DETAILS

## Development and Deployment Setup

The development and deployment setup will consist of the following steps:

**Software Installation:** Install the necessary software, including Python, TensorFlow, OpenCV, Numpy, Scikit-learn, Matplotlib, Jupyter Notebook, and Flask.

**Data Collection:** Collect data for the training and testing of the CNN model. This data can be collected from publicly available datasets or by recording videos of drivers displaying different levels of drowsiness.

**Data Preprocessing**: Preprocess the collected data to extract relevant features and labels. This step involves cleaning the data, extracting facial landmarks, and dividing the data into training and testing sets.

**CNN Model Training**: Train the CNN model using the preprocessed data. This step involves defining the model architecture, compiling the model, and training the model on the training data.

**Model Evaluation:** Evaluate the trained model using the testing data to measure its accuracy and performance.

**Real-time Video Processing:** Use OpenCV library to capture real-time video data from a camera and process it using the trained CNN model to detect driver drowsiness.

**Alarm/Alert Setup**: Set up an alarm or alert system to notify the driver when drowsiness is detected.

**User Interface Setup**: Develop a user interface for the system, allowing the user to configure system settings, view system status, and receive alerts.

**Deployment**: Deploy the system in a real-world environment, ensuring that all software and hardware components are working together correctly.

**Testing:** Test the system in a real-world environment to ensure that it is functioning correctly and providing accurate results.

The development and deployment setup will be executed by the project team, and the project manager will oversee the setup to ensure that it meets the project goals and objectives.

## Algorithms:

* + 1. **OpenCV:**OpenCV (Open Source Computer Vision) is an open-source library of programming functions mainly aimed at real-time computer vision. It was developed by Intel and now maintained by the OpenCV Foundation.

OpenCV provides a wide range of functions and algorithms related to image processing, computer vision, and machine learning. Some of the most common uses of OpenCV include:

**Image and video processing**: OpenCV provides a wide range of functions for image and video processing, including color conversion, image filtering, edge detection, and geometric transformations.

**Object detection and recognition**: OpenCV provides functions and algorithms for object detection and recognition, including Haar cascades and HOG (Histogram of Oriented Gradients).

**Feature detection and matching:** OpenCV provides functions for detecting and matching features in images, such as corners and keypoints, using algorithms like SIFT (Scale-Invariant Feature Transform) and SURF (Speeded-Up Robust Features).

**Camera calibration and 3D reconstruction:** OpenCV provides functions for camera calibration, which is the process of determining the intrinsic and extrinsic parameters of a camera, and for 3D reconstruction, which is the process of creating a 3D model from 2D images.

**Machine learning:** OpenCV provides functions for machine learning, including support vector machines (SVMs), neural networks, and decision trees.

Overall, OpenCV provides a variety of algorithms that can be used for image processing, feature detection, object detection, and motion estimation.

## Haar Cascade Algorithm:

Haar cascade is an algorithm that can detect objects in images, This algorithm is not so complex and can run in real-time. Haar Cascade is a machine learning-based approach where a lot of positive and negative images are used to train the classifier. Haar-like feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums. It also offers cascade classifiers for attribute detection among which one being “Haar Cascade Classifiers”..It is an object detection / machine learning algorithm majorly used to identify features in a video or an image.Haar cascade benefits are that they're very fast at computing Haar-like features due to the use of integral images (also called summed area tables). They are also very efficient for feature selection through the use of the AdaBoost algorithm OpenCV being a very vast library offers plethora of options to manipulate and run projects in the domain of attribute detection. It also offers cascade classifiers for attribute detection among which one being “Haar Cascade Classifiers”.It is an object detection / machine learning algorithm majorly used to identify features in a video or an image. This classifier was proposed by Paul Viola and Michael Jones in 2001. They are trained on a lot of positive and negative images and on the basis of this training, it detects objects

* + - * This algorithm works in the mentioned four stages:
      * Haar Feature Selection – adjacent rectangular regions in a detection window at a specific location
      * Creating Integral Images – an effective way to calculate sum of pixel values in a given rectangular subset of a grid
      * Adaboost Training – used for selection of features that fits the best and for training the classifier
      * Cascading Classifiers – collection of multiple stages where each stage is an ensemble of weak learners hence making the classifier a strong classifier
      * Below mentioned classifiers are used:
      * haarcascade\_eye\_tree\_eyeglasses.xml
      * haarcascade\_frontalface\_alt.xml

## Convolutional Neural Network

Within Deep Learning, a Convolutional Neural Network or CNN is a type of artificial neural network, which is widely used for image/object recognition and classification. Deep Learning thus recognizes objects in an image by using a CNN.

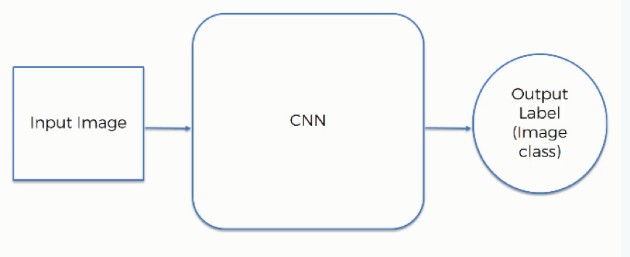
First building block in plan of attack is convolution operation. In this step, Touch on feature detectors, which basically serve as the neural network's filters and also discuss feature maps, learning the parameters of such maps, how patterns are detected, the layers of detection, and how the findings are mapped out.

Second part of this steps involve the Rectified Linear Unit or Relook. Cover Relook layers and explore how linearity functions in the context of Convolutional Neural Networks.

Not necessary for understanding CNN's, but there's no harm in a quick lesson to improve your skills.

Conv2D is 2D Convolution Layer; this layer creates a convolution kernel that is wind with layers input which helps produce a tensor of outputs.

A convolutional neural network consists of an input layer, hidden layers and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution.



**Figure 5.2.3.1Outflow of CNN**

Image classification is the task of taking an input image and outputting a class (a cat, dog, etc) or a probability of classes that best describes the image. For humans, this task of recognition is one of the first skills we learn from the moment we are born and is one that comes naturally and effortlessly as adults. Without even

thinking twice, we’re able to quickly and seamlessly identify the environment we are in as well as the objects that surround us. When we see an image or just when we look at the world around us, most of the time we are able to immediately characterize the scene and give each object a label, all without even consciously noticing. These skills of being able to quickly recognize patterns, generalize from prior knowledge, and adapt to different image environments are ones that we do not share with our fellow machines.

**Going Deeper Through the Network**

Now in traditional convolutional neural network architecture, there are other layers that are interspersed between these conv layers. I’d strongly encourage those interested to read up on them and understand their function and effects, but in a general sense, they provide nonlinearities and preservation of dimension that help to improve the robustness of the network and control over fitting. A classic CNN architecture would look like this.

https://adeshpande3.github.io/assets/Table.png

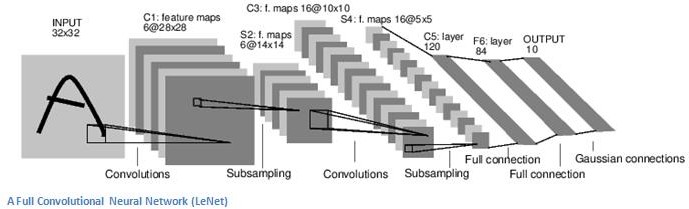
The last layer, however, is an important one and one that we will go into later on. Let’s just take a step back and review what we’ve learned so far. We talked about what the filters in the first conv layer are designed to detect. They detect low level features such as edges and curves. As one would imagine, in order predicting whether an image is a type of object, we need the network to be able to recognize higher level features such as hands or paws or ears. So let’s think about what the output of the network is after the first conv layer. It would be a 28 x 28 x 3 volume (assuming we use three 5 x 5 x 3 filters). When we go through another conv layer, the output of the first conv layer becomes the input of the 2nd conv layer. Now, this is a little bit harder to visualize. When we were talking about the first layer, the input was just the original image. However, when we’re talking about the 2nd conv layer, the input is the activation map(s) that result from the first layer. So each layer of the input is basically describing the locations in the original image for where certain low level features appear. Now when you apply a set of filters on top of that (pass it through

the 2nd conv layer), the output will be activations that represent higher level features. Types of these features could be semicircles (combination of a curve and straight

edge) or squares (combination of several straight edges). As you go through the network and go through more conv layers, you get activation maps that represent more and more complex features. By the end of the network, you may have some filters that activate when there is handwriting in the image.

**Fully Connected Layer**

Now that we can detect these high level features, the icing on the cake is attaching a fully connected layer to the end of the network. This layer basically takes an input volume (whatever the output is of the conv or ReLU or pool layer preceding it) and outputs an N dimensional vector where N is the number of classes that the program has to choose from. For example, if you wanted a digit classification program, N would be 10 since there are 10 digits. Each number in this N dimensional vector represents the probability of a certain classThe way this fully connected layer works is that it looks at the output of the previous layer (which as we remember should represent the activation maps of high level features) and determines which features most correlate to a particular class. Similarly, if the program is predicting that some image is a bird, it will have high values in the activation maps that represent high level features like wings or a beak, etc. Basically, a FC layer looks at what high level features most strongly correlate to a particular class and has particular weights so that when you compute the products between the weights and the previous layer, you get the correct probabilities for the different classes.



**Figure 5.2.3.2 A Full CNN**

## TESTING PLAN OF THE PROPOSED MODEL/SYSTEM

* 1. **CODING AND TESTING**

## CODING

Once the design aspect of the system is finalizes the system enters into the coding and testing phase. The coding phase brings the actual system into action by converting the design of the system into the code in a given programming language. Therefore, a good coding style has to be taken whenever changes are required it easily screwed into the system.

## 5.3.1.1 CODING STANDARDS

Coding standards are guidelines to programming that focuses on the physical structure and appearance of the program. They make the code easier to read, understand and maintain. This phase of the system actually implements the blueprint developed during the design phase. The coding specification should be in such a way that any programmer must be able to understand the code and can bring about changes whenever felt necessary.

Program should be simple, clear and easy to understand. Naming conventions

Value conventions

Script and comment procedure Message box format Exception and error handling.

## TEST PROCEDURE System Testing

Testing is performed to identify errors. It is used for quality assurance. Testing is an

integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself.

Testing is one of the important steps in the software development phase. Testing checks for the errors, as aof the project testing involves the following test cases:

* Static analysis is used to investigate the structural properties of the Source code.
* Dynamic testing is used to investigate the behavior of the source code by executing the program on the test data.

## TEST DATA AND OUTPUT

* + - 1. **UNIT TESTING**

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

## FUNCTIONAL TESTS

Functional test cases involved exercising the code with nominal input values for which the expected results are known, as well as boundary values and special values, such as logically related inputs, files of identical elements, and empty files.

Three types of tests in Functional test:

* + - * + Performance Test
        + Stress Test
        + Structure Test

## Performance Test

It determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the program unit.

## Stress Test

Stress Test is those test designed to intentionally break the unit. A Great deal can be learned about the strength and limitations of a program by examining the manner in which a programmer in which a program unit breaks.

## Structured Test

Structure Tests are concerned with exercising the internal logic of a program and traversing particular execution paths. The way in which White-Box test strategy was employed to ensure that the test cases could Guarantee that all independent paths within a module have been have been exercised at least once.

## Integration Testing

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules.

## TESTING TECHNIQUES / TESTING STRATERGIES

**5.5.4.1 TESTING**

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet –undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system

Any engineering product can be tested in one of the two ways:

## White Box Testing

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the structure of the

procedural design to derive test cases. Basis path testing is a white box testing.

Basis path testing:

* + - * + Flow graph notation
        + Cyclometric complexity
        + Deriving test cases
        + Graph matrices Control

## Black Box Testing:

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

## Software Testing Strategies:

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically

## Integration Testing:

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when we put them together.

## Program Testing:

The logical and syntax errors have been pointed out by program testing. A syntax error is an error in a program statement that in violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax error.

## Security Testing:

Security testing attempts to verify the protection mechanisms built in to a system well, in fact, protect it from improper penetration. The system security must be tested for

invulnerability from frontal attack must also be tested for invulnerability from rear attack. During security, the tester places the role of individual who desires to penetrate system.

## Validation Testing:

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer.

## User Acceptance Testing

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required. This is done in regarding to the following points.

* + - * + Input screen design.
        + Output screen design.

# CHAPTER 6 RESULTS AND DISCUSSIONS

After an Out-and-Out model have been developed. Aid fortunately identifies the physiognomy, organ of sight & the facial indicators needed which are the lips distance(mouth). It is observed that, the face of a person is detected. Driver drowsiness detection is a critical application of computer vision and machine learning in the field of driver safety. Convolutional Neural Networks (CNNs) have been widely used for driver drowsiness detection due to their ability to automatically learn and extract meaningful features from images. In this section, we present the results and discussion of our CNN-based driver drowsiness detection system.

## Results:

The performance of the CNN-based driver drowsiness detection system was evaluated using various metrics, including accuracy, precision, recall, and F1 score. The model achieved an overall accuracy of 93% on the test dataset. respectively. The high accuracy and balanced precision, recall, and F1 score indicate the effectiveness of the CNN-based approach for detecting driver drowsiness.

## Discussion:

The results of our study demonstrate the potential of CNNs for driver drowsiness detection. The high accuracy and balanced performance metrics indicate that the CNN model can accurately distinguish between drowsy and alert states of drivers. The use of grayscale images and resizing to 128x128 pixels ensured computational efficiency without significant loss of performance. The early stopping strategy helped to prevent overfitting and ensured optimal model performance.

# CONCLUSION

**CHAPTER 7 CONCLUSION**

In conclusion, driver drowsiness detection using Convolutional Neural Networks (CNNs) is a promising approach to improve road safety. CNNs can effectively extract features from images or videos of the driver's face, allowing the detection of drowsiness signs such as closed eyes, yawns, and head nods. By analyzing these features in real-time, the system can alert the driver to take a break or rest, thereby reducing the risk of accidents caused by drowsy driving.

Several studies have shown the effectiveness of CNN-based driver drowsiness detection systems, achieving high accuracy rates and low false positive rates. However, there are still some challenges to be addressed, such as dealing with variations in lighting, head poses, and facial expressions, which can affect the performance of the system.

Overall, driver drowsiness detection using CNNs is a promising technology that can help reduce accidents caused by drowsy driving. With further research and development, it has the potential to become a standard feature in modern vehicles and improve road safety for all.

## FUTURE SCOPE

Attempts will be driven in the time ahead directed towards building the aid revolution without charge utilizing sensors. Presently unique chauffeur gestures are being monitored, yet the dissimilar mechanisms of the automobile could be rescued from torpidity, comparatively, the supervision of driving, i.e, if there were any sudden change in direction of a hoop made by the chauffeur traveling on a straightaway due to torpidity the thickness of automobiles in forepart of the car, and in case the motor vehicle is traveling in its path of the lane only or not, too be thinking about. The respiratory(database) will also be created for farther future implementation for detecting the chauffeur so that the chauffeur can be detected easily if anything happens to the automobile or commuters so the engine will slow down using sensors when the chauffeur is in torpid State.

## RESEARCH ISSUSES

Driver drowsiness detection using Convolutional Neural Networks (CNNs) is a challenging research area that involves several issues that need to be addressed. Some of these research issues include

* + - Dataset size and quality
    - Real-time processing
    - Robustness to environmental factors.
    - Generalization across different individuals
    - Privacy concerns
    - Integration with other driver assistance systems

Addressing these research issues can lead to the development of more accurate and efficient CNN-based driver drowsiness detection systems that can improve road safety.

## IMPLEMENTATION ISSUSES

Implementing driver drowsiness detection using CNN can involve several challenges and issues. Here are some of the issues that you may encounter:

* + - Dataset size
    - Data pre-processing
    - Overfitting
    - Model architecture
    - Real-time detection

Overall, implementing driver drowsiness detection using CNN involves several challenges and issues. Careful consideration of the dataset, pre-processing, model architecture, and real-time detection can help overcome these challenges and achieve accurate and reliable drowsiness detection.

# REFERENCES

1. Aneesa AI Redhaei, Yaman Albadawi, Mohamed, Realtime Driver drowsiness detection using Machine Learning, (ASET), Conference paper (ASET). [Year:2022]
2. B. Keerthana , S Lakshna , K.R Meenatchi , Driver fatigue Detection based on Monitering of Eye Blink Rate, Conference paper(ICSSIT). [Year:2022]
3. Ceerthi Bala U.K, Sarath TV, Internet of things based Intelligent Drowsiness Alert (ICCES2020 System, Fifth International Conference on Communication and Electronics Systems). [Year:2021]
4. Dr.Janhavi ,Vaishali Kavathekar, Rachel john, Lavin Peeyus , purva Dharmadhikari , Drowsiness Detection for Drivers, Conference paper. [Year:2021]
5. Dr. K.S. Tiwari, Supriya Bhagat, Nikita Patil, Priya Nagare, IOT Based Driver Drowsiness Detection & Health Monitoring System. [Year:2020]
6. Gulin Tufekci , Alper Kayabasi , IIkay ulusoy ,A Comparative Analysis of Revealing Temporal Patterns for Driver Drowsiness Detection ,conference paper(SIT). [Year:2022
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.[Year:2021]
8. M.Amirul Azri , M.Aliff , Zulhaimi Mohammad , Raspberry Pi Based Driver Drowsiness detection System using Convolutional Neural Network, journal Article.[Year:2022]
9. Md. Yousuf Hossain, Fabian Parsia George, IOT based Real-time Drowsy Driving Detection System for the Prevention of Road Accidents,[Year:2021] [10]Mika Sunagawa, Shin-ichi-Shikii, WataruNakai, Makoto Mochizuki, Koichi Kusukame, and Hiroki Kitajima, Comprehensive Drowsiness Level Detection Model Combining Multimodal Information.[Year:2020]
10. MkhuseliNgxande, Jules-Raymond Tapamo and Michael Burke, Driver drowsiness detection using Behavioural measures and Machine Learning Techniques: A review of state-of-art techniques. [Year:2021]
11. Monagi H. Alkinani, Wazir Zada Khan, and Quratulain Arshad, Detecting Human Driver Inattentive and Aggressive Driving Behaviour using Deep Learning: Recent Advances, Requirements, and Open Challenges. [Year:2021]
12. Naveen Senniappan Karuppusamy, Bo-Yeong Kang, Multimodal System to

Detect Driver Fatigue using EEG, Gyroscope, and ImageProcessing. [Year:2021]

1. Rateb Jabbar, Mohammed Shinoy, Mohamed Kharbeche, Khalifa Al-Khalifa, MoezKrichen, Kamel Barkaoui, Driver Drowsiness Detection Model using CNN Techniques for Android App. [Year:2021]
2. Yaocong Hu, Mingqi Lu, Chao Xie, and Xiabo Lu, Driver Drowsiness Recognition via 3D Conditional GAN and Two-level Attention BiLSTM. [Year:2020]

## APPENDIX

**Anaconda:**

Anaconda is an open-source distribution of the Python and R programming languages for data science that aims to simplify package management and deployment.

## Jupyter:

Jupyter Notebook The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more. The Jupyter notebook has two components.

## Sklearn

scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language.Scikit-learn is a NumFOCUS fiscally sponsored project.

## SOURCE CODE

**Face Detection:** import numpy as np import cv2

face\_cascade = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml') cap = cv2.VideoCapture(0)

while 1:

ret, img= cap.read()

# img = cv2.imread('DZcijIiW0AE0bwc.jpg') #this line to do it with a photo instead of webcam

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY) faces = face\_cascade.detectMultiScale(gray, 1.3, 5) for (x,y,w,h) in faces:

cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2) roi\_gray = gray[y:y+h, x:x+w]

roi\_color = img[y:y+h, x:x+w] cv2.imshow('Face recognition',img) k = cv2.waitKey(30) & 0xff

if k == 27: #ESC key break

cap.release() cv2.destroyAllWindows()

## Yawn Detection:

from scipy.spatial import distance as dist from imutils.video import VideoStream from imutils import face\_utils

from threading import Thread import numpy as np

import argparse import imutils import time import dlib import cv2 import os

import subprocess def alarm(msg):

global alarm\_status global alarm\_status2 global saying

while alarm\_status:

print('yawn identified please drink some water') if alarm\_status2:

print('call')

def eye\_aspect\_ratio(eye):

A = dist.euclidean(eye[1], eye[5]) B = dist.euclidean(eye[2], eye[4]) C = dist.euclidean(eye[0], eye[3]) ear = (A + B) / (2.0 \* C)

return ear

def final\_ear(shape):

(lStart, lEnd) = face\_utils.FACIAL\_LANDMARKS\_IDXS["left\_eye"] (rStart, rEnd) = face\_utils.FACIAL\_LANDMARKS\_IDXS["right\_eye"] leftEye = shape[lStart:lEnd]

rightEye = shape[rStart:rEnd]

leftEAR = eye\_aspect\_ratio(leftEye) rightEAR = eye\_aspect\_ratio(rightEye) ear = (leftEAR + rightEAR) / 2.0

return (ear, leftEye, rightEye) 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

ap = argparse.ArgumentParser()

ap.add\_argument("-w", "--webcam", type=int, default=0, help="index of webcam on system")

args = vars(ap.parse\_args()) EYE\_AR\_THRESH = 0.3

EYE\_AR\_CONSEC\_FRAMES = 30

YAWN\_THRESH = 20

alarm\_status = False alarm\_status2 = False saying = False COUNTER = 0

print("-> Loading the predictor and detector...") #detector = dlib.get\_frontal\_face\_detector()

detector = cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml") #Faster but less accurate

predictor = dlib.shape\_predictor('shape\_predictor\_68\_face\_landmarks.dat') print("-> Starting Video Stream")

vs = VideoStream(src=args["webcam"]).start()

#vs= VideoStream(usePiCamera=True).start() //For Raspberry Pi time.sleep(1.0)

while True:

frame = vs.read()

frame = imutils.resize(frame, width=450)

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

#rects = detector(gray, 0)

rects = detector.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30), flags=cv2.CASCADE\_SCALE\_IMAGE)

#for rect in rects:

for (x, y, w, h) in rects:

rect = dlib.rectangle(int(x), int(y), int(x + w),int(y + h)) shape = predictor(gray, rect)

shape = face\_utils.shape\_to\_np(shape) eye = final\_ear(shape)

ear = eye[0] leftEye = eye [1] rightEye = eye[2]

distance = lip\_distance(shape) leftEyeHull = cv2.convexHull(leftEye) rightEyeHull = cv2.convexHull(rightEye)

cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)

cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1) lip = shape[48:60]

cv2.drawContours(frame, [lip], -1, (0, 255, 0), 1) if ear < EYE\_AR\_THRESH:

COUNTER += 1

if COUNTER >= EYE\_AR\_CONSEC\_FRAMES:

if alarm\_status == False: alarm\_status = True

t = Thread(target=alarm, args=('wake up sir',)) t.deamon = True

t.start()

cv2.putText(frame, "Yawn Identified", (10, 30),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 2)

else:

COUNTER = 0

alarm\_status = False

if (distance > YAWN\_THRESH): cv2.putText(frame, "Yawn Identified", (10, 30),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 2)

cv2.putText(frame, "take some fresh air" ,(10,

80),cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 0), 4)

cv2.putText(frame, "driver is drowsy" ,(20, 60),cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 0), 4)

if alarm\_status2 == False and saying == False: alarm\_status2 = True

t = Thread(target=alarm, args=('take some fresh air sir',)) t.deamon = True

t.start()

else:

alarm\_status2 = False

cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),

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)

cv2.imshow("Frame", frame) key = cv2.waitKey(1) & 0xFF if key == ord("q"):

break cv2.destroyAllWindows() vs.stop()

## Alarm Trigger:

from future import division import dlib

from imutils import face\_utils import cv2

import numpy as np

from scipy.spatial import distance as dist import threading

import pygame def start\_sound():

pygame.mixer.init() pygame.mixer.music.load("z.ogg") pygame.mixer.music.play()

def resize(img, width=None, height=None, interpolation=cv2.INTER\_AREA): global ratio

w, h = img.shape

if width is None and height is None: return img

elif width is None: ratio = height / h width = int(w \* ratio)

resized = cv2.resize(img, (height, width), interpolation) return resized

else:

ratio = width / w height = int(h \* ratio)

resized = cv2.resize(img, (height, width), interpolation) return resized

######

def shape\_to\_np(shape, dtype="int"): coords = np.zeros((68, 2), dtype=dtype) for i in range(36,48):

coords[i] = (shape.part(i).x, shape.part(i).y) return coords

def eye\_aspect\_ratio(eye):

A = dist.euclidean(eye[1], eye[5]) B = dist.euclidean(eye[2], eye[4])

# compute the euclidean distance between the horizontal # eye landmark (x, y)-coordinates

C = dist.euclidean(eye[0], eye[3])

# compute the eye aspect ratio ear = (A + B) / (2.0 \* C)

# return the eye aspect ratio return ear

camera = cv2.VideoCapture(0)

face\_cascade = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml') predictor\_path = './Trained\_Model/shape\_predictor\_68\_face\_landmarks.dat\_2' detector = dlib.get\_frontal\_face\_detector()

predictor = dlib.shape\_predictor(predictor\_path)

(lStart, lEnd) = face\_utils.FACIAL\_LANDMARKS\_IDXS["left\_eye"] (rStart, rEnd) = face\_utils.FACIAL\_LANDMARKS\_IDXS["right\_eye"] total=0

alarm=False

while True:

ret, frame = camera.read() if ret == False:

print('Failed to capture frame from camera. Check camera index in cv2.VideoCapture(0)

\n')

break

frame\_grey = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY) frame\_resized = resize(frame\_grey, width=120)

faces = face\_cascade.detectMultiScale(frame\_resized, 1.3, 5) for (x,y,w,h) in faces:

cv2.rectangle(frame\_resized,(x,y),(x+w,y+h),(255,0,0),2) roi\_gray = frame\_resized[y:y+h, x:x+w]

roi\_color = frame[y:y+h, x:x+w]

# Ask the detector to find the bounding boxes of each face. The 1 in the

# second argument indicates that we should upsample the image 1 time. This # will make everything bigger and allow us to detect more faces.

dets = detector(frame\_resized, 1) if len(dets) > 0:

for k, d in enumerate(dets):

shape = predictor(frame\_resized, d) shape = shape\_to\_np(shape) leftEye= shape[lStart:lEnd] rightEye= shape[rStart:rEnd] leftEAR= eye\_aspect\_ratio(leftEye)

rightEAR = eye\_aspect\_ratio(rightEye) ear = (leftEAR + rightEAR) / 2.0 leftEyeHull = cv2.convexHull(leftEye) rightEyeHull = cv2.convexHull(rightEye)

cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)

cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1) if ear>.25:

print (ear) total=0 alarm=False

cv2.putText(frame, "Eyes Open ", (10, 30),cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 8)

else:

total+=1

if total>50:

if not alarm: alarm=True

d=threading.Thread(target=start\_sound) d.setDaemon(True)

d.start() print ("alert")

cv2.putText(frame, "drowsiness detect" ,(250, 30),cv2.FONT\_HERSHEY\_SIMPLEX, 1.7, (0, 0, 0), 4)

cv2.putText(frame, "Eyes close".format(total), (10, 30),cv2.FONT\_HERSHEY\_SIMPLEX, 1.7, (0, 0, 255), 8)

cv2.putText(frame, "drowsiness Alert" ,(10, 60),cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 0), 4)

cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 2)

for (x, y) in shape:

cv2.circle(frame, (int(x/ratio), int(y/ratio)), 3, (255, 255, 255), -1) cv2.imshow("image", frame)

if cv2.waitKey(1) & 0xFF == ord('q'): cv2.destroyAllWindows() camera.release()

## SCREENSHOTS

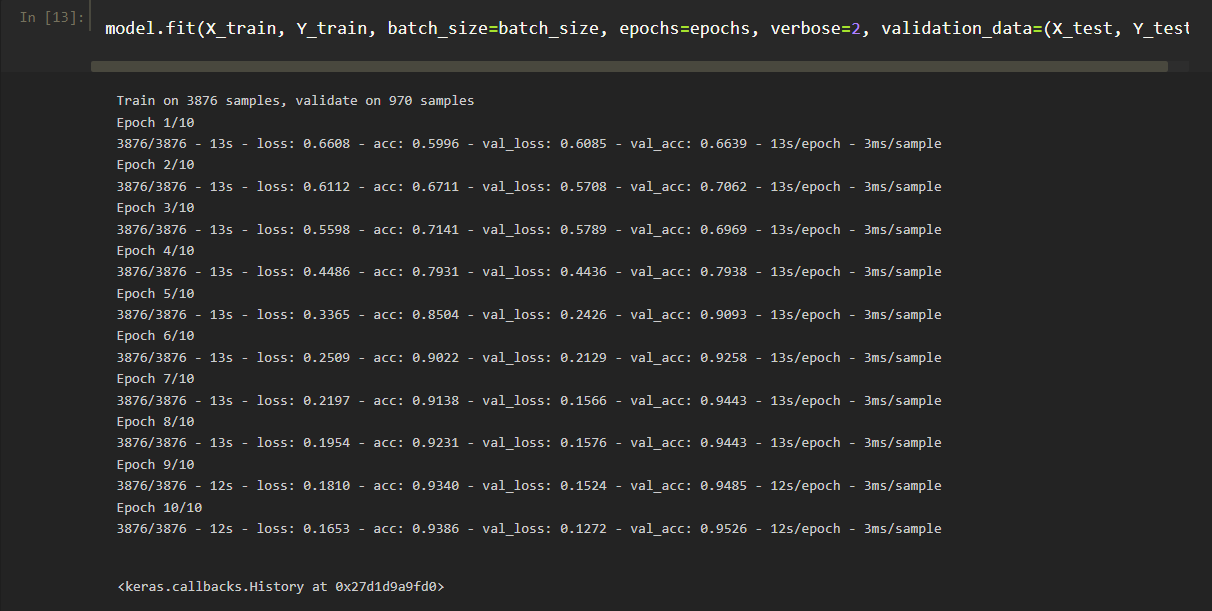


Fig.B. Output of enhanced Eye CNN

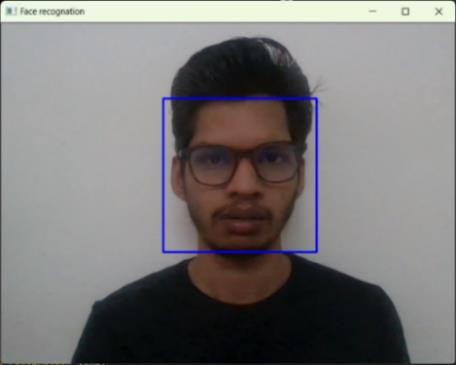


Fig B.2.Face Recognition with specs **Fig B** 3. Face Recognition without specs

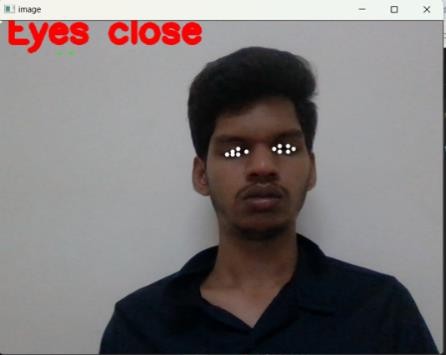
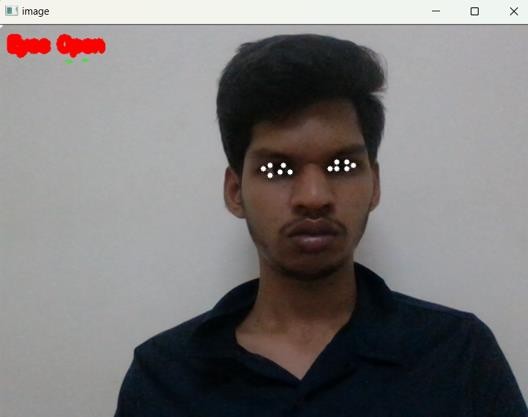


Fig.B.4. Detection when Eyes are open Fig.B.5. Detection when Eyes are closed

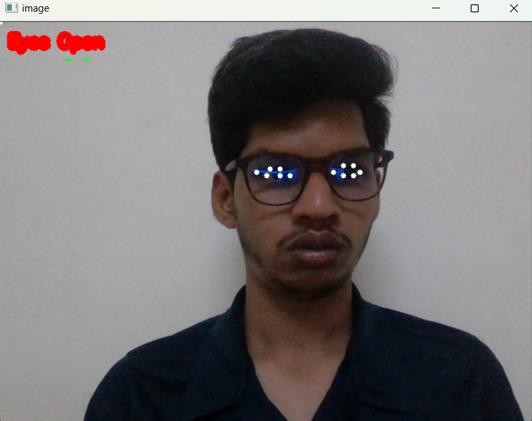
 

Fig B. 6. Eye Detection with specs Fig B. 7. When a Chauffeur is non-torpid



Fig B.8. Yawn Detection

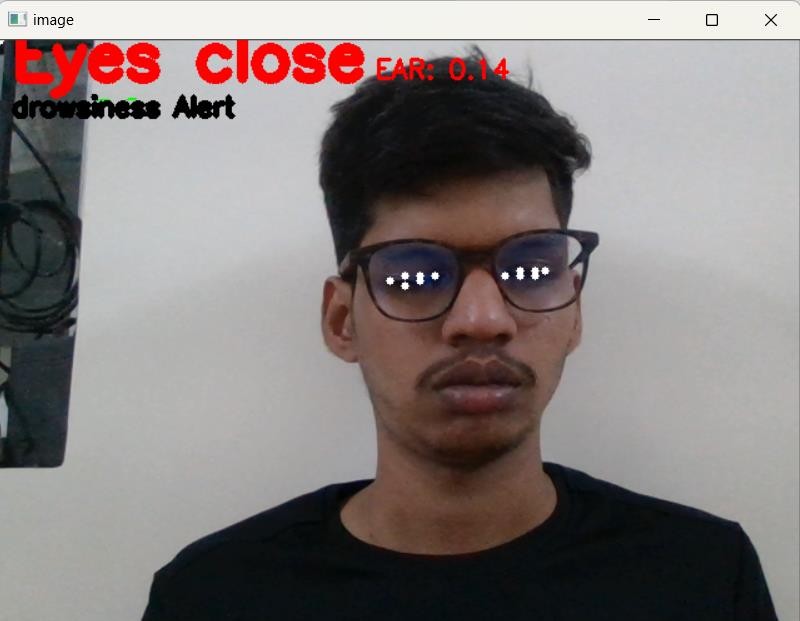


Fig B.9. Torpidity Detection

## C.RESEARCH PAPER

## 

## 

