**Driver Somnolence Detection with Alert System using Advance Computer Vision**

Submitted in partial fulfillment of the requirements for the award of

Bachelor of Engineering degree in Computer Science and Engineering

By

**Sahithi Prathyusha (Reg. No - 40110476)**

**S.L.N.D. Pavan (Reg.no – 40111286)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

## SCHOOL OF COMPUTING

SATHYABAMA

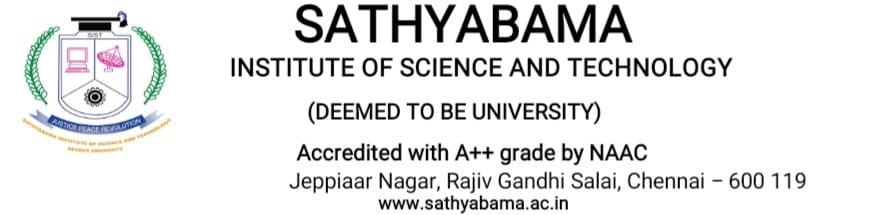
**INSTITUTE OF SCIENCE AND TECHNOLOGY (DEEMED TO BE UNIVERSITY)**

**Accredited with Grade “A” by NAAC**

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600119**

## NOVEMBER - 2023

i



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**BONAFIDE CERTIFICATE**

This is to certify that this Project Report is the bonafide work of **Sahithi prathyusha(40110476) and S.L.N.D.Pavan(40111286)** who carried out the Project Phase-1 entitled **“Driver Somnolence Detection with Alert system using Advance computer vision”** under my supervision from june 2023 to November 2023.

**Internal Guide**

**Dr. Ms. T. Anandhi, M.E.,( Ph.D).,**

**Head of the Department Dr. L .Lakshmanan ,M.E.,( Ph.D).,**

**Submitted for Viva voce Examination held on**

**Internal Examiner External Examiner**

ii

**DECLERATION**

I, **Sahithi prathyusha (Reg. No- 40110476),** hereby declare that the Project Phase-1 Report entitled **Driver Somnolence Detection with Alert system using Advance computer Vision”** done by me under the guidance of **Dr.Ms.T.Anandhi,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:**

**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, and **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.Ms.T.Anandhi M.E.,(Ph.D.,)** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase-1 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**

The advance of computing technology has provided the means for building intelligent vehicle systems. The driver drowsiness detection system is one of the potential applications of intelligent vehicle systems. Here we employ machine learning techniques to detect driver drowsiness. The input is given in the form of a real-time webcam video, which will be processed in real-time.

The system gives an output in the form of a red alert and also a sound alert will be shown if the user is found closing their eyes. The proposed system aims to lessen the number of accidents that occur due to drivers’ drowsiness and fatigue, which will in turn increase transportation safety. The existing system leverages CNN which accounts for its low accuracy and high processing time. The proposed system leverages eye aspect ratio and eyepoints are used to detect the output perfectly. The face, an important part of the body, conveys a lot of information. When a driver is in a state of fatigue, the facial expressions, e.g., the frequency of blinking and yawning, are different from those in the normal state. In this paper, we propose a system that detects the drivers’ fatigue status, such as yawning, blinking, and duration of eye closure, using video images, without equipping their bodies with devices. Owing to the shortcomings of previous algorithms, we introduce a new face-tracking algorithm to improve the tracking accuracy. Further, we designed a new detection method for facial regions based on 68 key points. Then we use these facial regions to evaluate the drivers’ state. By combining the features of the eyes and mouth. The system can alert the driver using a fatigue warning. The experimental results showed that the system achieved high accuracy

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **TITLE** | **PAGE NO** |
|  | ABSTRACT | **v** |
|  | LIST OF FIGURES | **viii** |
|  | LIST OF ABBREVIATIONS | **vii** |
| **1** | **INTRODUCTION** | **1** |
| **2** | **LITERATURE SURVEY** | **3** |
| **3** | **SYSTEM ANALYSIS** | **6** |
|  | 3.1 EXISTING SYSTEM | **6** |
|  | 3.2 PROPOSED SYSTEM | **7** |
|  | 3.3 REQUIREMENT SPECIFICATION | **8** |
|  | 3.4 PLATFORM SPECIFICATION | **8** |
| **4** | **SYSTEM DESIGN** | **13** |
|  | 4.1SYSTEM ARCHITECTURE | **13** |
|  | 4.2 USE-CASE DIAGRAM | **13** |
|  | 4.3 ACTIVITY DIAGRAM | **14** |
|  | 4.4 SEQUENCE DIAGRAM | **15** |
|  | 4.5 DATA FLOW DIAGRAM 0 | **16** |
|  | 4.6 DATA FLOW DIAGRAM 1 | **16** |
|  | 4.7 ER DIAGRAM | **17** |
| **5** | **MODULE DESCRIPTION** | **19** |
| **6** | **TESTING** | **22** |
|  | 6.1 SYSTEM TESTING | **22** |
|  | 6.2. TESTING TECHNIQUES | **24** |
|  | 6.3 SOFTWARE TESTING STRATEGIES | **26** |
| **7** | **CONCLUSION** | **29** |
|  | **APPENDIX 1** | **30** |
|  | **APPENDIX 2** | **46** |
|  | **REFERENCES** | **48** |

## LIST OF ACRONYMS AND ABBREVIATIONS

CNN - Convolutional Neural Network YOLO - You Only Look Once

DL - Deep Learning

## LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **TITLE** | **PAGE NO** |
| 4.1 | SYSTEM ARCHITECTURE | **13** |
| 4.2 | USE-CASE DIAGRAM | **14** |
| 4.3 | ACTIVITY DIAGRAM | **15** |
| 4.4 | SEQUENCE DIAGRAM | **16** |
| 4.5 | DATA FLOW DIAGRAM 0 | **16** |
| 4.6 | DATA FLOW DIAGRAM 1 | **17** |
| 4.7 | ER DIAGRAM | **18** |

**CHAPTER 1 INTRODUCTION**

**1.1 INTRODUCTION**

Somnolence and sleepiness of drivers is an important cause of road accidents on expressways, highways, and motorways. These accidents not only result in an economic loss but may also result in physical injuries, which could result in permanent disability or even death.

In recent years, the fatigue-driving-detection system has become a hot research topic. The detection methods are categorized 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, evaluation, and ﬁlling in questionnaires. Then, these data are used to estimate the vehicles being driven by tired drivers, assisting the drivers to plan their schedules accordingly. However, drivers’ feedback is not required in the objective detection method as it monitors the driver’s physiological state and driving-behavior characteristics in real-time [4]. 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, non-contact is cheaper and more convenient because the system that does not require computer Vision technology or a sophisticated camera allows the use of the device in more cars.

The aim of this research is to minimize this cause of road accidents. The safe driving requirement is unavoidable and to attain this, a driver’s drowsiness detection system is to be incorporated into vehicles. Somnolence detection using a vehicle-based, physiological, and behavioral change measurement system is possible with embedded pros and cons. Advancements in the field of image processing and the development of faster and cheaper processors direct researchers to focus on behavioral change measurement systems for Somnolence detection.

Computer vision-based drowsiness detection is possible by closely monitoring the drowsiness symptoms like eye blinking intervals, yawning, eye closing duration, head position etc. The project deals with the merits and demerits of the

drowsiness symptoms measurement mechanism and computer vision-based drowsiness detection systems.

The conclusion of the research is that by designing a hybrid computer vision-based drowsy driver detection system dependability is achieved. The proposed system is non-intrusive in nature and helpful in attaining safer roads by limiting potential accidental threats due to driver drowsiness.

## CHAPTER 2 LITERATURE REVIEWS

The proposed approach in (1) Drunkenness or exhaustion is a leading cause of car accidents, with severe implications for road safety. More fatal accidents could be avoided if fatigued drivers were warned ahead of time. Several drowsiness detection technologies to monitor for signs of inattention while driving and notifying the driver can be adopted.

In (2) Detection of drowsiness of driver is a vehicle safety technology, which helps to put off accidents which caused by the driver being dozy. A variety of studies have recommended that around 20% of all road accidents are due to drowsiness of the driver. The developments of technologies for detecting or preventing drowsiness while driving is a major confront in accident evasion systems. Because of the peril of the tiredness while driving, different new methods need to be developed for counteracting the effect.

In (3), We propose drowsiness detection in real-time surveillance videos by determining if a person’s eyes are open or closed. As a first step, the face of the subject is detected in the image. In the detected face, the eyes are localized and filtered with an extended Sobel operator to detect the curvature of the eyelids. Once the curves are detected, concavity is used to tell whether the eyelids are closed or open.

However, the proposed study in (4) This paper presents a study in which driver's gaze zone is categorized using new deep learning techniques. Since the sequence of gaze zones of a driver reflects precisely what and how he behaves, it allows us infer his drowsiness, focusing or distraction by analyzing the images coming from a camera.

According to (5), This paper presents an accurate method of drowsiness detection for the images obtained using low resolution consumer grade web cameras under normal lighting conditions. The drowsiness detection method uses Haar based cascade classifier for eye tracking and combination of Histogram of oriented gradient (HOG) features.

In (6) Drowsiness and Fatigue of drivers are amongst the significant causes of road accidents. Every year, they increase the amounts of deaths and fatalities injuries globally. In this paper, a module for Advanced Driver Assistance System (ADAS) is presented to reduce the number of accidents due to drivers fatigue and hence increase the transportation safety.

Similarly, the proposed study in (7) Drowsiness has been the significant cause of horrible accidents which is causing deaths and fatalities injuries. Day by Day fatal injuries numbers are increasing globally. From the past many years, researchers have concluded drivers with a lack of sleep and more tiredness which causes drowsiness of the driver. this paper shows a new experimental model is designed.

In (8) first, Drowsy driving is one of the reasons for automobile accidents. We propose a ‘Driver Drowsiness Detection System’ which can help to reduce automobile accidents caused due to drowsy driving. We propose a Convolutional Neural Network (CNN) model that is capable of detecting drowsiness.

The proposed study in (9) number of automobile accidents due to driver drowsiness is increasing at an alarming rate. An automated non-contact system that can identify driver drowsiness early is the need of the hour. Motivated by this alarming need, a novel method is proposed that can detect driver drowsiness at an early stage and avoid mishaps..

Finally in (10), driver drowsiness and sleep are significant cause of road accidents, especially when drivers drive for a long time on highways. Avoiding

an accident can be the aim of smart systems nowadays. A robust driver detection system must be designed to alert the driver.

## CHAPTER 3 SYSTEM ANALYSIS

* 1. **EXISTING SYSTEM**

Different techniques have been reported for the detection of driver drowsiness. Although detection results achieved are promising, these traditional approaches are still far from being highly accurate and efficient. The existing systems are simple and effective but are extremely vulnerable to impact. The existing systems are based on multiple calculations and are highly complex in nature. Due to high complexity and inefficiency, the CNN-based models are inaccurate and have low precision. The existing system also leverages the Raspberry Pi Camera, found to be ineffective during the late hours and in the nighttime.

**Demerits of the existing system:**

* + - The limitations of the existing model are that the Raspberry Pi camera was used and because of this, the system could not work during the nighttime. A night-vision camera should have been used.
    - Also, the dataset used was small and improper, which led to low accuracy and low precision.
    - The existing model leverages CNN while the technology in transport devices has now advanced towards computer vision.
    - In the existing system, a fixed blink duration is assumed even though everyone’s blink duration lasts differently.
    - It is observed that the system's accuracy decreases in bad lighting conditions.
    - There is no alert sound module in the existing system.
  1. **PROPOSED SYSTEM**

The system being proposed is a driver drowsiness detection system that can be placed in a vehicle. The proposed real-time algorithm will estimate the eye aspect ratio that measures the eye open level in each video frame. It perceives eye-blink patterns as EAR values. By doing so, potential drowsiness is detected. The blink patterns differ mostly in the speed of closing and the speed of opening, the degree of squeezing an eye, and in duration. We propose to exploit the facial landmark detection model for the localization of the eyes and their contours.

The objective of the paper is to detect driver drowsiness by analyzing human eye blinks using a recent facial landmark detection and to make use of E.A.R(eye aspect ratio) for easy, fast, and efficient blink detection. Firstly, the face is localized in the image using facial landmark detection. Then, shape prediction methods are used to detect important features on the face.

Facial landmark detection can be implemented by using a training image set of labeled facial landmarks. The set is manually labeled by marking (x,y) coordinates of regions of interest.

The Eye aspect ratio is used and eyepoints are used to detect the output perfectly.

**Advantages of the proposed system:**

* + - The results showed that the system was successful in driver drowsiness detection by providing a reliably precise enough estimation of the level of eye openness.
    - This alert system can be used in real-time due to a very negligible performance cost experienced in facial landmark detection.
    - The results are highly accurate.
    - Even if the user wears specs, we are able to detect the output and an alert sound module is added to the proposed system.
    - The system is highly accurate, precise, lightweight and takes less time to process.
    - The model has many real-world applications as it can be used for reducing road accidents, and help the driver gain focus and take full control of his driving.
  1. **REQUIREMENT SPECIFICATION**
     1. HARDWARE REQUIREMENTS

Processor : Pentium Dual Core 2.00GHZ

Hard disk : 120 GB

RAM : 2GB (minimum)

Keyboard : 110 keys enhanced

* + 1. SOFTWARE REQUIREMENTS

Operating system : Windows7 (with service pack 1), 8, 8.1 and 10 IDE : Anaconda

Language : Python

* 1. **PLATFORM SPECIFICATION – ANACONDA**

Anaconda is an open-source package manager for Python and R. It is the most popular platform among data science professionals for running Python and R implementations. There are over 300 libraries in data science, so having a robust

distribution system for them is a must for any professional in this field. Anaconda simplifies package deployment and management. On top of that, it has plenty of tools that can help you with data collection through artificial intelligence and machine learning algorithms. With Anaconda, you can easily set up, manage, and share Conda environments. Moreover, you can deploy any required project with a few clicks when you’re using Anaconda. There are many advantages to using Anaconda and the following are the most prominent ones among them: Anaconda is free and open-source. This means you can use it without spending any money. In the data science sector, Anaconda is an industry staple. It is open-source too, which has made it widely popular. If you want to become a data science professional, you must know how to use Anaconda for Python because every recruiter expects you to have this skill. It is a must-have for data science.

It has more than 1500 Python and R data science packages, so you don’t face any compatibility issues while collaborating with others. For example, suppose your colleague sends you a project which requires packages called A and B but you only have package A. Without having package B, you wouldn’t be able to run the project. Anaconda mitigates the chances of such errors. You can easily collaborate on projects without worrying about any compatibility issues. It gives you a seamless environment that simplifies deploying projects. You can deploy any project with just a few clicks and commands while managing the rest. Anaconda has a thriving community of data scientists and machine learning professionals who use it regularly. If you encounter an issue, chances are, the community has already answered the same. On the other hand, you can also ask people in the community about the issues you face there, it’s a very helpful community ready to help new learners. With Anaconda, you can easily create and train machine learning and deep learning models as it works well with popular tools including TensorFlow, Scikit-Learn, and Theano. You can create visualizations by using Bokeh, Holoviews, Matplotlib, and Datashader while using Anaconda.

**How to Use Anaconda for Python**

Now that we have discussed all the basics in our Python Anaconda tutorial, let’s discuss some fundamental commands you can use to start using this package manager.

Listing All Environments

To begin using Anaconda, you’d need to see how many Conda environments are present in your machine.

conda env list

It will list all the available Conda environments in your machine. Creating a New Environment

You can create a new Conda environment by going to the required directory and use this command:

conda create -n <your\_environment\_name>

You can replace <your\_environment\_name> with the name of your environment. After entering this command, conda will ask you if you want to proceed to which you should reply with y:

proceed ([y])/n)?

On the other hand, if you want to create an environment with a particular version of Python, you should use the following command:

conda create -n <your\_environment\_name> python=3.6

Similarly, if you want to create an environment with a particular package, you can use the following command:

conda create -n <your\_environment\_name>pack\_name

Here, you can replace pack\_name with the name of the package you want to use. If you have a .yml file, you can use the following command to create a new Conda environment based on that file:

conda env create -n <your\_environment\_name> -f <file\_name>.yml

We have also discussed how you can export an existing Conda environment to a

.yml file later in this article.

**Activating an Environment**

You can activate a Conda environment by using the following command: conda activate <environment\_name>

You should activate the environment before you start working on the same. Also, replace the term <environment\_name> with the environment name you want to activate. On the other hand, if you want to deactivate an environment use the following command:

conda deactivate

**Installing Packages in an Environment**

Now that you have an activated environment, you can install packages into it by using the following command:

conda install <pack\_name>

Replace the term <pack\_name> with the name of the package you want to install in your Conda environment while using this command.

**Updating Packages in an Environment**

If you want to update the packages present in a particular Conda environment, you should use the following command:

conda update

The above command will update all the packages present in the environment. However, if you want to update a package to a certain version, you will need to use the following command:

conda install <package\_name>=<version>

**Exporting an Environment Configuration**

Suppose you want to share your project with someone else (colleague, friend,

etc.). While you can share the directory on Github, it would have many Python packages, making the transfer process very challenging. Instead of that, you can create an environment configuration .yml file and share it with that person. Now, they can create an environment like your one by using the .yml file.

For exporting the environment to the .yml file, you’ll first have to activate the same and run the following command:

conda env export ><file\_name>.yml

The person you want to share the environment with only has to use the exported file by using the ‘Creating a New Environment’ command we shared before.

**Removing a Package from an Environment**

If you want to uninstall a package from a specific Conda environment, use the following command:

conda remove -n <env\_name><package\_name>

On the other hand, if you want to uninstall a package from an activated environment, you’d have to use the following command:

conda remove <package\_name>

**Deleting an Environment**

Sometimes, you don’t need to add a new environment but remove one. In such cases, you must know how to delete a Conda environment, which you can do so by using the following command:

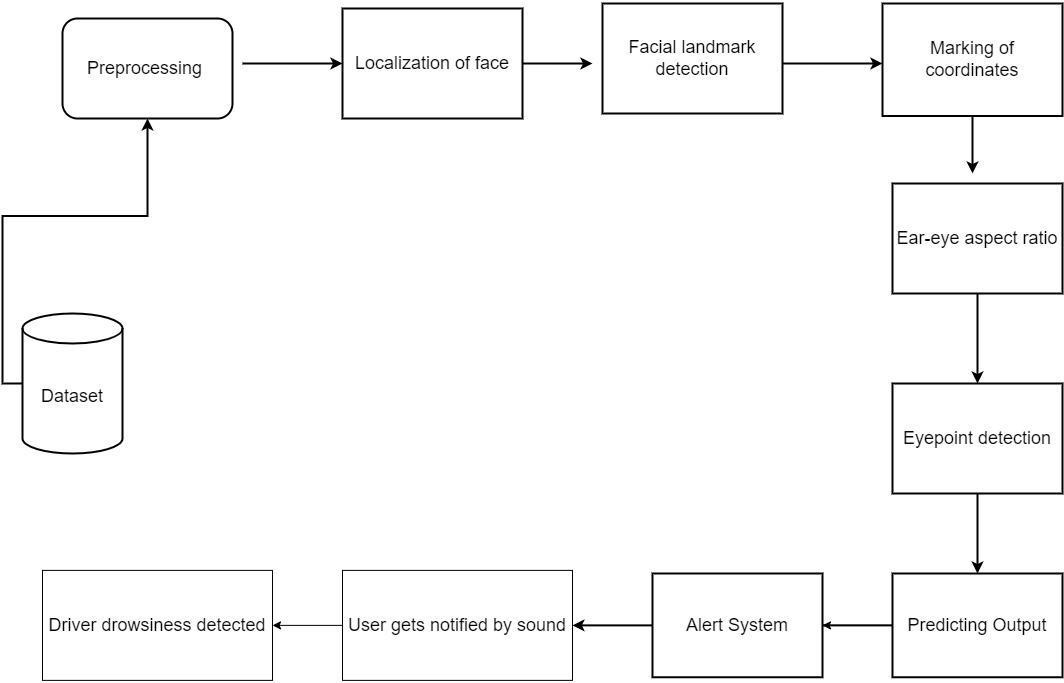
conda env remove –name <env\_name>

The above command would delete the Conda environment right away.

## CHAPTER 4 SYSTEM DESIGN

* 1. **SYSTEM ARCHITECTURE**

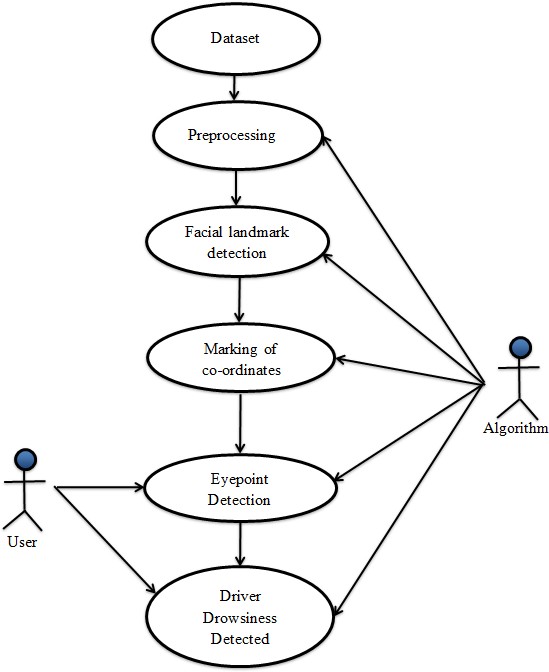
This diagram is nothing but a simple description of all the entities that have been incorporated into the system. The diagram represents the relations between each of them and involves a sequence of decision-making processes and steps. You can simply call it a visual or the whole process and its implementation. All functional correspondences are explained in this diagram.



**Fig 4.1 – Architecture Diagram**

* 1. **USE-CASE DIAGRAM**

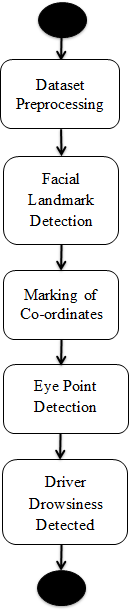
A use case diagram is a standard diagram that shows all interactions between the user, dataset, and algorithm used. It is developed in the early stages of the process.



**Fig 4.2 – Use-Case Diagram**

* 1. **ACTIVITY DIAGRAM**

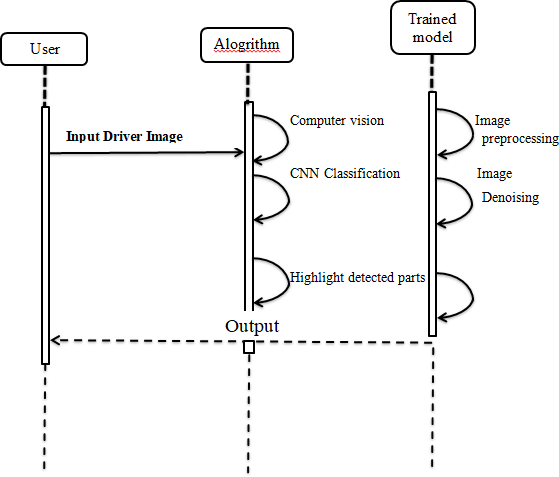
In simple terms, a diagram that represents the order of all activities is called the activity diagram. It shows the workflow between different activities that take place in the whole process. However, these are not exactly flowcharts but are similar.



**Fig 4.3 – Activity Diagram**

* 1. **SEQUENCE DIAGRAM**

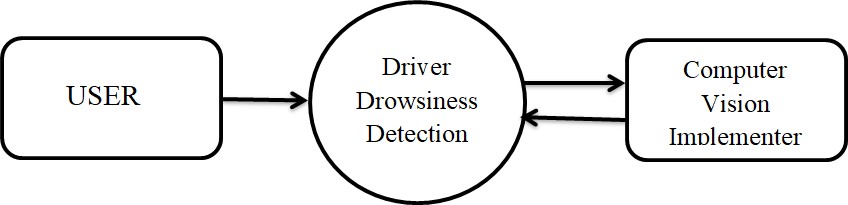
These are other kinds of interaction-based diagrams that show how all the operations are carried out. They capture the context of collaborations between objects and processes.



**Fig 4.4 – Sequence Diagram**

* 1. **DATA FLOW DIAGRAM LEVEL 0**

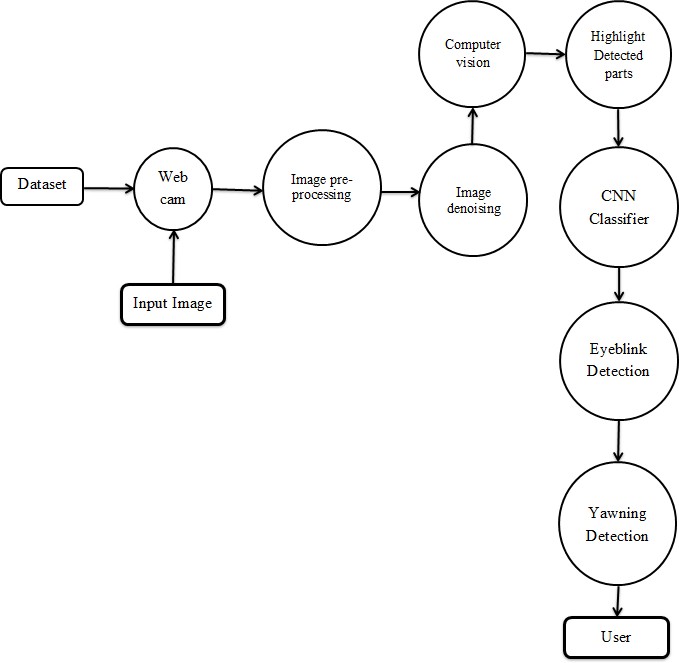
This is basically a contextual diagram, also referred to as a “context diagram”. It only represents the top level or the 0 Level in the whole process. it gives an abstraction kind of view and shows the whole system as a single process and its relationship to externalities.



**Fig 4.5 – Data Flow Diagram 0**

* 1. **DATA FLOW DIAGRAM LEVEL 1**

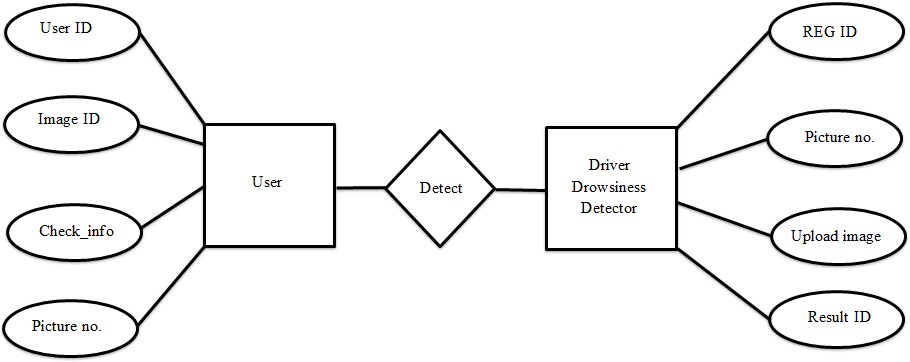
Level DFDs represent the complete system as a single process. it notates every process and sub-process that comes together in a sequence to form the complete system. This along with ) and 2-level data flow diagrams comprise the “fundamental system model”.



**Fig 4.6 – Data Flow Diagram 1**

* 1. **ER DIAGRAM**

ER stands for Entity Relationship. These diagrams display the relationship of entities that are used and stored in the database. They explain the structure of the whole process. these diagrams can be made using three basic concepts, attributed, relationships, and entities.



**Fig 4.7 – ER Diagram**

## CHAPTER 5 MODULE DESCRIPTION

**MODULE DESCRIPTION**

* Module 1: Dataset training and pre-processing
* Module 2: Building a prediction model using YOLO
* Module 3: Prediction of drowsiness and implementation of alarm module
  1. **MODULE 1 : DATASET TRAINING AND PRE-PROCESSING**

When a lot of data is available everywhere, improper examination of analyzing data might result in misleading conclusions. Thus, before performing any analysis, the representation and quality of data must come first. Data preprocessing is the process of alteration or removal of data before being utilized for some purpose. This process assures or improves performance, and it is a crucial stage in the data mining process. Data cleaning is the first step of data preprocessing in data mining.

Another method is ‘smoothing’ the data by using regression. Regression may be linear or multiple, but the motive is to render the data smooth enough for a trend to be visible.

Data Transformation: A popular way is a normalization. In this approach, every point of data is subtracted from the highest value of data in that field and then divided by the range of data in that field.

Data Reduction: This reduces the amount of data through the following techniques and makes it easier to analyze.

* 1. **MODULE 2: Building a prediction model using YOLO**

The YOLO model expects the training images to be of the same dimensions. The first step in labeling is drawing bounding boxes around the objects within the image and providing labels for them. The second step is to choose which format those dimensions will take. SAS supports the following 3 formats for bounding boxes – YOLO, RECT and COCO. The last step is merging each image’s XML file with the image file to create a new image that contains these bounding boxes. Since the “YOLO” format normalizes bounding box coordinates between 0 and 1, it is not necessary to resize your images even if they are not of the same dimensions. You simply mention the dimensions that you want for your resized image. For the YOLO model to adjust to the varying sizes of objects, we need to generate anchor boxes. For every prediction grid, we define a preset number of anchor boxes of predefined shapes and sizes. There will be one anchor box per grid that is responsible for predicting the object whose center lies in that grid.

architecture for the YOLO model:

* + - The default image size in the documentation is 416x416 pixels. Adjust this parameter to match your image size.
    - Predictions are made by dividing an image into square grids each of size 32x32 pixels. Using your image size, determine the size of your prediction grids by dividing your image size by 32.
    - The “anchors” parameter should be set to the list of anchor box values that were derived using the get\_anchors() method.
    - There are two parameters within the method that often get confused, max\_labels\_per\_image and max\_label. The difference between them is:
    - max\_label\_per\_image: this denotes the maximum number of bounding box labels that appear in an image in the training set. For example, if you have

3 images with 2 objects each and 1 image with 5 objects, this parameter would be set to 5.

* + - max\_labels: The maximum number of bounding box predictions that you want the model to predict per test image.

## 3: PREDICTION OF DROWSINESS AND IMPLEMENTATION OF ALARM MODULE

YOLO, as we know, predicts a single object per grid cell. While this makes the built model simpler, it creates issues when a single cell has more than one object, as YOLO can only assign a single class to the cell. YOLOv2 gets rid of this limitation by allowing the prediction of multiple bounding boxes from a single cell. Two-stage object detection refers to the use of algorithms that break down the object detection problem statement into the following two stages:

* + 1. Detecting possible object regions.
    2. Classifying the image in those regions into object classes.

Popular two-step algorithms like Fast-RCNN and Faster-RCNN typically use a Region Proposal Network that proposes regions of interest that might contain objects. The output from the RPN is then fed to a classifier that classifies the regions into classes.

While this gives accurate results in object detection with a high mean Average Precision (mAP), it results in multiple iterations taking place in the same image, thus slowing down the detection speed of the algorithm and preventing real-time detection.

## CHAPTER 6 TESTING

**TESTING METHODOLOGIES**

There are many different types of testing methods or techniques used as part of the software testing methodology. Some of the important testing methodologies are:

* 1. **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. For example, the design must not have any logic faults in the design is detected before coding commences, otherwise, the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as a walkthrough.

Testing is one of the important steps in the software development phase. Testing checks for the errors, as a whole of 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.

**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.

**Structure 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.

* Exercise all logical decisions on their true or false sides.
* Execute all loops at their boundaries and within their operational bounds.
* Exercise internal data structures to assure their validity.
* Checking attributes for their correctness.
* Handling end of file conditions, I/O errors, buffer problems, and textual errors in the output information

**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 interfacing. i.e., integration testing is the complete testing of the set of modules that makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach evolved from the unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that interface dispenses can be easily found and corrected.

The major error that was faced during the project is a linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and its intercommunications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

* 1. **TESTING TECHNIQUES / TESTING STRATEGIES**

Testing is the 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. It verifies that the whole set of programs hangs together. System testing requires a test consisting of several key activities and steps for running a program, string, and system and is important in adopting a successful new system.

This is the last chance to detect and correct errors before the system is installed for user acceptance testing.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise, the program or the project is not said to be complete. Software testing is the critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing is the process of executing the program with the intent of finding the error. A good test case design is one that has a probability of finding a yet undiscovered error. A successful test is one that uncovers a yet undiscovered error. Any engineering product can be tested in one of the two ways:

**White-box testing**

This testing is also called Glass box testing. In this testing, by knowing the specific functions that a product has been designed to perform tests can be conducted that demonstrate each function is fully operational and at the same time search for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is 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, a 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.

The steps involved in black-box test case design are:

* + Graph-based testing methods
  + Equivalence partitioning
  + Boundary value analysis
  + Comparison testing
  1. **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. For this reason, a template for software testing a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics:

* Testing begins at the module level and works “outward” toward the integration of the entire computer-based system.
* Different testing techniques are appropriate at different points in time.
* The developer of the software and an independent test group conducts testing.
* Testing and Debugging are different activities but debugging must be accommodated in any testing strategy.

**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 it. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when we put them together. The problem, of course, is “putting them together”- interfacing. There may be the chances of data loss across another’s sub-functions when combined may not produce the desired major function; individually acceptable impressions may be magnified to unacceptable levels; global data structures can present problems.

**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 violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax errors. These errors are shown through error messages generated by the computer. A logic error on the other hand deals with the incorrect data fields, out-off-range items, and invalid combinations. Since the compiler s will not deduct logical errors, the programmer must examine the output. Condition testing exercises the logical conditions contained in a module. The possible types of elements in a condition include a Boolean operator, Boolean variable, a pair of Boolean parentheses A relational operator or an arithmetic expression. The condition testing method focuses on testing each condition in the program the purpose the of condition test is to deduct not only errors in the condition of a program but also other errors in the program.

**Security Testing**

Security testing attempts to verify the protection mechanisms built into a system well, in fact, protect it from improper penetration. The system security must be tested for invulnerability from frontal attacks must also be tested for invulnerability from rear attacks. During security, the tester places the role of the individual who desires to penetrate the system.

**Validation Testing**

At the culmination of integration testing, the 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 a manner that is reasonably expected by the customer.

Software validation is achieved through a series of black-box tests that demonstrate conformity with requirements. After the validation test has been conducted, one of two conditions exists.

* The function or performance characteristics conform to specifications and are accepted.
* A validation from the specification is uncovered and a deficiency is created.

Deviation or errors discovered at this step in this project are corrected prior to completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies. Thus the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily. Though there were deficiencies in the system they were not catastrophic.

**User Acceptance Testing**

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

* Input screen design.
* Output screen design.

## CHAPTER 7

**CONCLUSION AND FUTURE ENHANCEMENTS**

The results showed that the system was successful in identifying driver fatigue by providing an accurate estimate of the quantity of eye-opening. This warning system can be used in real-time due to the exceptionally cheap performance cost associated with facial landmark detection. The outcomes are extremely exact. We can recognise the output even if the user is wearing glasses and add an alarm sound module into the suggested system. The technology is extremely precise, lightweight, and requires very little processing time. The notion has a number of practical applications, including reducing road accidents and supporting drivers in recovering full focus and control of their vehicles.

## APPENDIX 1

rom sys import flags

from scipy.spatial import distance from imutils import face\_utils import imutils

import dlib import cv2

import numpy as np

## Yawn Functions

def get\_landmarks(im): rects = detect(im, 1)

if len(rects) > 1: return "error"

if len(rects) == 0: return "error"

return np.matrix([[p.x, p.y] for p in predict(im, rects[0]).parts()])

def annotate\_landmarks(im, landmarks): im = im.copy()

for idx, point in enumerate(landmarks): pos = (point[0, 0], point[0, 1]) cv2.putText(im, str(idx), pos,

fontFace=cv2.FONT\_HERSHEY\_SCRIPT\_SIMPLEX,

fontScale=0.4, color=(0, 0, 255))

cv2.circle(im, pos, 3, color=(0, 255, 255)) return im

def top\_lip(landmarks): top\_lip\_pts = []

for i in range(50,53): top\_lip\_pts.append(landmarks[i])

for i in range(61,64): top\_lip\_pts.append(landmarks[i])

top\_lip\_all\_pts = np.squeeze(np.asarray(top\_lip\_pts)) top\_lip\_mean = np.mean(top\_lip\_pts, axis=0)

return int(top\_lip\_mean[:,1])

def bottom\_lip(landmarks): bottom\_lip\_pts = []

for i in range(65,68): bottom\_lip\_pts.append(landmarks[i])

for i in range(56,59): bottom\_lip\_pts.append(landmarks[i])

bottom\_lip\_all\_pts = np.squeeze(np.asarray(bottom\_lip\_pts)) bottom\_lip\_mean = np.mean(bottom\_lip\_pts, axis=0)

return int(bottom\_lip\_mean[:,1])

def mouth\_open(image):

landmarks = get\_landmarks(image)

if landmarks == "error": return image, 0

image\_with\_landmarks = annotate\_landmarks(image, landmarks) top\_lip\_center = top\_lip(landmarks)

bottom\_lip\_center = bottom\_lip(landmarks) lip\_distance = abs(top\_lip\_center - bottom\_lip\_center) return image\_with\_landmarks, lip\_distance

## Yawn Functions end

def eye\_aspect\_ratio(eye):

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

return ear

#

yawns = 0 yawn\_status = False #

thresh = 0.25

frame\_check = 20

detect = dlib.get\_frontal\_face\_detector() predict =

dlib.shape\_predictor("models/yolo\_shape\_estimation\_2017xGPU2019.dat")

(lStart, lEnd) = face\_utils.FACIAL\_LANDMARKS\_68\_IDXS["left\_eye"] (rStart, rEnd) = face\_utils.FACIAL\_LANDMARKS\_68\_IDXS["right\_eye"] cap=cv2.VideoCapture(0)

flag=0 flag\_yawn = 0

while True:

ret, frame=cap.read()

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

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

Array

#

image\_landmarks, lip\_distance = mouth\_open(frame) #

subjects = detect(gray, 0) for subject in subjects:

shape = predict(gray, subject)

shape = face\_utils.shape\_to\_np(shape)#converting to NumPy

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)

# lips contour

mouth = shape[48:68]

mouthHULL = cv2.convexHull(mouth)

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

if ear < thresh:

flag += 1

# print (flag)

if flag >= frame\_check: cv2.putText(frame,

"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ALERT!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*", (10, 30),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0,

255), 2)

cv2.putText(frame,

"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ALERT!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*", (10,325),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0,

255), 2)

else:

#print ("Drowsy") flag = 0

#yawn

prev\_yawn\_status = yawn\_status if lip\_distance > 25:

yawn\_status = True

cv2.putText(frame, "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ALERT!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*", (10, 30),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0,

255), 2)

output\_text = " Yawn Count: " + str(yawns + 1)

cv2.putText(frame, output\_text, (50,50),

cv2.FONT\_HERSHEY\_COMPLEX,

1,(0,255,127),2)

print(yawns)

else:

yawn\_status = False

if prev\_yawn\_status == True and yawn\_status == False: yawns += 1

#yawn end

# #how all lables

#cv2.imshow("Labelled face", image\_landmarks)

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

break cv2.destroyAllWindows()

cap.release()

from functools import wraps

import numpy as np import tensorflow as tf

from keras import backend as K

from keras.layers import Conv2D, MaxPool2D, Add, ZeroPadding2D,

UpSampling2D, Concatenate

from keras.layers.advanced\_activations import LeakyReLU from keras.layers.normalization import BatchNormalization from keras.regularizers import l2

from keras.models import Model, Input

def NetworkConv2D\_BN\_Leaky(input, channels, kernel\_size, kernel\_regularizer = l2(5e-4), strides=(1,1), padding="same", use\_bias=False):

network = Conv2D( filters=channels, kernel\_size=kernel\_size, strides=strides, padding=padding, kernel\_regularizer=kernel\_regularizer, use\_bias=use\_bias)(input)

network = BatchNormalization()(network) network = LeakyReLU(alpha=0.1)(network) return network

def residual\_block(input, channels, num\_blocks): network = ZeroPadding2D(((1,0), (1,0)))(input)

network = NetworkConv2D\_BN\_Leaky(input=network,channels=channels, kernel\_size=(3,3), strides=(2,2), padding="valid")

for blocks in range(num\_blocks):

network\_1 = NetworkConv2D\_BN\_Leaky(input=network, channels= channels // 2, kernel\_size=(1,1))

network\_1 = NetworkConv2D\_BN\_Leaky(input=network\_1,channels= channels, kernel\_size=(3,3))

network = Add()([network, network\_1]) return network

def darknet(input):

network = NetworkConv2D\_BN\_Leaky(input=input, channels=32, kernel\_size=(3,3))

network = residual\_block(input=network, channels=64, num\_blocks=1) network = residual\_block(input=network, channels=128, num\_blocks=2) network = residual\_block(input=network, channels=256, num\_blocks=8) network = residual\_block(input=network, channels=512, num\_blocks=8) network = residual\_block(input=network, channels=1024, num\_blocks=4)

return network

def last\_layers(input, channels\_in, channels\_out, layer\_name=""):

network = NetworkConv2D\_BN\_Leaky( input=input, channels=channels\_in, kernel\_size=(1,1))

network = NetworkConv2D\_BN\_Leaky(input=network, channels= (channels\_in \* 2) , kernel\_size=(3, 3))

network = NetworkConv2D\_BN\_Leaky(input=network, channels=channels\_in, kernel\_size=(1, 1))

network = NetworkConv2D\_BN\_Leaky(input=network, channels=(channels\_in \* 2), kernel\_size=(3, 3))

network = NetworkConv2D\_BN\_Leaky(input=network, channels=channels\_in, kernel\_size=(1, 1))

network\_1 = NetworkConv2D\_BN\_Leaky(input=network, channels=(channels\_in \* 2), kernel\_size=(3, 3))

network\_1 = Conv2D(filters=channels\_out, kernel\_size=(1,1), name=layer\_name)(network\_1)

return network, network\_1

def yolo\_main(input, num\_anchors, num\_classes): darknet\_network = Model(input, darknet(input))

network, network\_1 = last\_layers(darknet\_network.output, 512, num\_anchors

* (num\_classes + 5), layer\_name="last1")

network = NetworkConv2D\_BN\_Leaky( input=network, channels=256, kernel\_size=(1,1))

network = UpSampling2D(2)(network)

network = Concatenate()([network, darknet\_network.layers[152].output])

network, network\_2 = last\_layers(network, 256, num\_anchors \* (num\_classes + 5), layer\_name="last2")

network = NetworkConv2D\_BN\_Leaky(input=network, channels=128, kernel\_size=(1, 1))

network = UpSampling2D(2)(network)

network = Concatenate()([network, darknet\_network.layers[92].output])

network, network\_3 = last\_layers(network, 128, num\_anchors \* (num\_classes + 5), layer\_name="last3")

return Model(input, [network\_1, network\_2, network\_3])

def tiny\_yolo\_main(input, num\_anchors, num\_classes):

network\_1 = NetworkConv2D\_BN\_Leaky(input=input, channels=16, kernel\_size=(3,3) )

network\_1 = MaxPool2D(pool\_size=(2,2), strides=(2,2), padding="same")(network\_1)

network\_1 = NetworkConv2D\_BN\_Leaky(input=network\_1, channels=32, kernel\_size=(3, 3))

network\_1 = MaxPool2D(pool\_size=(2, 2), strides=(2, 2), padding="same")(network\_1)

network\_1 = NetworkConv2D\_BN\_Leaky(input=network\_1, channels=64, kernel\_size=(3, 3))

network\_1 = MaxPool2D(pool\_size=(2, 2), strides=(2, 2), padding="same")(network\_1)

network\_1 = NetworkConv2D\_BN\_Leaky(input=network\_1, channels=128, kernel\_size=(3, 3))

network\_1 = MaxPool2D(pool\_size=(2, 2), strides=(2, 2), padding="same")(network\_1)

network\_1 = NetworkConv2D\_BN\_Leaky(input=network\_1, channels=256, kernel\_size=(3, 3))

network\_2 = MaxPool2D(pool\_size=(2, 2), strides=(2, 2), padding="same")(network\_1)

network\_2 = NetworkConv2D\_BN\_Leaky(input=network\_2, channels=512, kernel\_size=(3, 3))

network\_2 = MaxPool2D(pool\_size=(2, 2), strides=(1, 1), padding="same")(network\_2)

network\_2 = NetworkConv2D\_BN\_Leaky(input=network\_2, channels=1024, kernel\_size=(3, 3))

network\_2 = NetworkConv2D\_BN\_Leaky(input=network\_2, channels=256, kernel\_size=(1, 1))

network\_3 = NetworkConv2D\_BN\_Leaky(input=network\_2, channels=512, kernel\_size=(3, 3))

network\_3 = Conv2D(num\_anchors \* (num\_classes + 5), kernel\_size=(1,1))(network\_3)

network\_2 = NetworkConv2D\_BN\_Leaky(input=network\_2, channels=128, kernel\_size=(1, 1))

network\_2 = UpSampling2D(2)(network\_2)

network\_4 = Concatenate()([network\_2, network\_1])

network\_4 = NetworkConv2D\_BN\_Leaky(input=network\_4, channels=256, kernel\_size=(3, 3))

network\_4 = Conv2D(num\_anchors \* (num\_classes + 5), kernel\_size=(1,1))(network\_4)

return Model(input, [network\_3, network\_4])

import tensorflow as tf

from keras import backend as K from PIL import Image

def yolo\_head(feats, anchors, num\_classes, input\_shape, calc\_loss=False): num\_anchors = len(anchors)

anchors\_tensor = K.reshape(K.constant(anchors), [1, 1, 1, num\_anchors, 2])

grid\_shape = K.shape(feats)[1:3]

grid\_y = K.tile(K.reshape(K.arange(0, stop=grid\_shape[0]), [-1, 1, 1, 1]),

[1, grid\_shape[1], 1, 1])

grid\_x = K.tile(K.reshape(K.arange(0, stop=grid\_shape[1]), [1, -1, 1, 1]),

[grid\_shape[0], 1, 1, 1])

grid = K.concatenate([grid\_x, grid\_y]) grid = K.cast(grid, K.dtype(feats))

feats = K.reshape(

feats, [-1, grid\_shape[0], grid\_shape[1], num\_anchors, num\_classes + 5])

box\_xy = (K.sigmoid(feats[..., :2]) + grid) / K.cast(grid\_shape[::-1], K.dtype(feats))

box\_wh = K.exp(feats[..., 2:4]) \* anchors\_tensor / K.cast(input\_shape[::-1], K.dtype(feats))

box\_confidence = K.sigmoid(feats[..., 4:5]) box\_class\_probs = K.sigmoid(feats[..., 5:])

if calc\_loss == True:

return grid, feats, box\_xy, box\_wh

return box\_xy, box\_wh, box\_confidence, box\_class\_probs

def yolo\_correct\_boxes(box\_xy, box\_wh, input\_shape, image\_shape): box\_yx = box\_xy[..., ::-1]

box\_hw = box\_wh[..., ::-1]

input\_shape = K.cast(input\_shape, K.dtype(box\_yx)) image\_shape = K.cast(image\_shape, K.dtype(box\_yx))

new\_shape = K.round(image\_shape \* K.min(input\_shape/image\_shape)) offset = (input\_shape-new\_shape)/2./input\_shape

scale = input\_shape/new\_shape box\_yx = (box\_yx - offset) \* scale box\_hw \*= scale

box\_mins = box\_yx - (box\_hw / 2.) box\_maxes = box\_yx + (box\_hw / 2.) boxes = K.concatenate([

box\_mins[..., 0:1],

box\_mins[..., 1:2],

box\_maxes[..., 0:1],

box\_maxes[..., 1:2]

])

boxes \*= K.concatenate([image\_shape, image\_shape]) return boxes

def yolo\_boxes\_and\_scores(feats, anchors, num\_classes, input\_shape, image\_shape):

box\_xy, box\_wh, box\_confidence, box\_class\_probs = yolo\_head(feats, anchors, num\_classes, input\_shape)

boxes = yolo\_correct\_boxes(box\_xy, box\_wh, input\_shape, image\_shape) boxes = K.reshape(boxes, [-1, 4])

box\_scores = box\_confidence \* box\_class\_probs box\_scores = K.reshape(box\_scores, [-1, num\_classes]) return boxes, box\_scores

def yolo\_eval(yolo\_outputs,

anchors, num\_classes, image\_shape, max\_boxes=20, score\_threshold=.6, iou\_threshold=.5):

num\_layers = len(yolo\_outputs)

anchor\_mask = [[6,7,8], [3,4,5], [0,1,2]] if num\_layers==3 else [[3,4,5],

[1,2,3]]

input\_shape = K.shape(yolo\_outputs[0])[1:3] \* 32 boxes = []

box\_scores = []

for l in range(num\_layers):

\_boxes, \_box\_scores = yolo\_boxes\_and\_scores(yolo\_outputs[l], anchors[anchor\_mask[l]], num\_classes, input\_shape, image\_shape)

boxes.append(\_boxes) box\_scores.append(\_box\_scores)

boxes = K.concatenate(boxes, axis=0) box\_scores = K.concatenate(box\_scores, axis=0)

mask = box\_scores >= score\_threshold

max\_boxes\_tensor = K.constant(max\_boxes, dtype='int32') boxes\_ = []

scores\_ = [] classes\_ = []

for c in range(num\_classes):

class\_boxes = tf.boolean\_mask(boxes, mask[:, c]) class\_box\_scores = tf.boolean\_mask(box\_scores[:, c], mask[:, c]) nms\_index = tf.image.non\_max\_suppression(

class\_boxes, class\_box\_scores, max\_boxes\_tensor, iou\_threshold=iou\_threshold)

class\_boxes = K.gather(class\_boxes, nms\_index) class\_box\_scores = K.gather(class\_box\_scores, nms\_index) classes = K.ones\_like(class\_box\_scores, 'int32') \* c boxes\_.append(class\_boxes) scores\_.append(class\_box\_scores)

classes\_.append(classes)

boxes\_ = K.concatenate(boxes\_, axis=0) scores\_ = K.concatenate(scores\_, axis=0) classes\_ = K.concatenate(classes\_, axis=0)

return boxes\_, scores\_, classes\_

def letterbox\_image(image, size): iw, ih = image.size

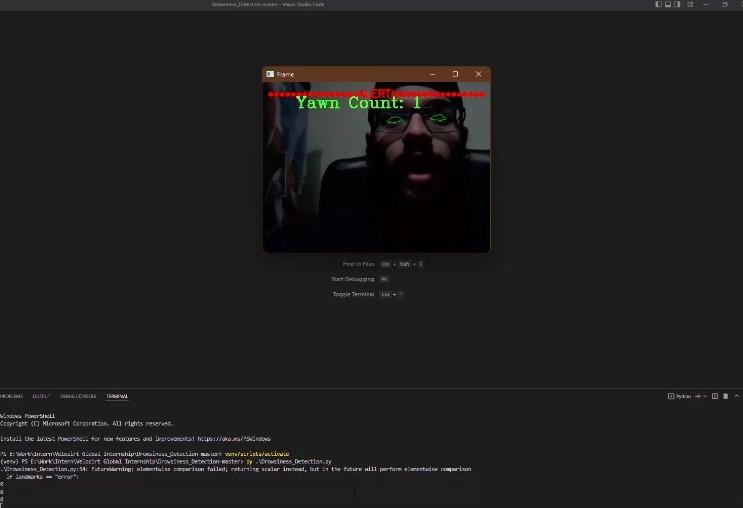
w, h = size

scale = min(w/iw, h/ih) nw = int(iw\*scale)

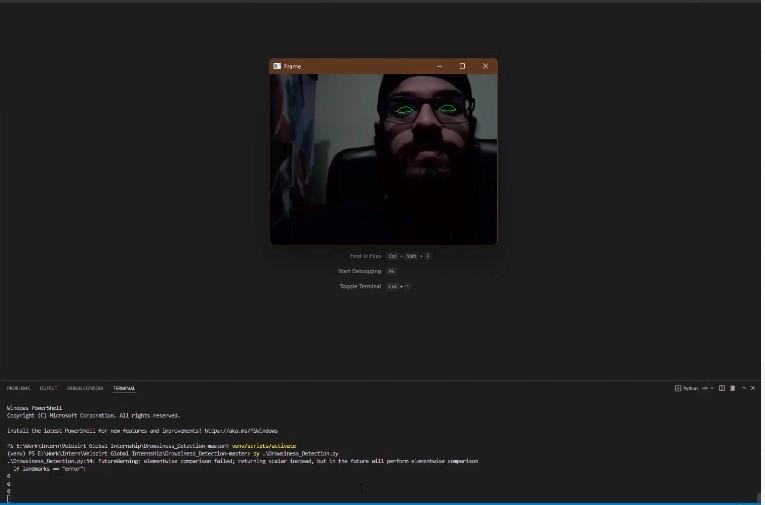
nh = int(ih\*scale)

image = image.resize((nw,nh), Image.BICUBIC) new\_image = Image.new('RGB', size, (128,128,128)) new\_image.paste(image, ((w-nw)//2, (h-nh)//2)) return new\_image

# APPENDIX 2



## Fig 9.1 Yawning detected



**Fig 9.2 Drowsiness detected**

# REFERENCES

1. AYMAN ALTAMEEM, 2019, “Early Identification and Detection of Driver Drowsiness by Hybrid Machine Learning,” ICICCT 2019: Intelligent Computing Paradigm and Cutting-edge Technologies pp 290-303, 2019
2. Subbarao, A., Sahithya, K. (2019) Driver Drowsiness Detection System for Vehicle Safety, International Journal of Innovative Technology and Exploring Engineering (IJITEE)
3. Tayab Khan, M., Anwar, H., Ullah, F., Ur Rehman, A., Ullah, R., Iqbal, A., … Kwak,

K. S. (2019). “Smart Real-Time Video Surveillance Platform for Drowsiness Detection Based on Eyelid Closure,” Wireless Communications and Mobile Computing, 2019

1. Choi, In-Ho, Sung Kyung Hong, and Yong-Guk Kim. ”Real-time categorization of driver’s gaze zone using the deep learning techniques.” 2016 International Conference on Big Data and Smart Computing (BigComp). IEEE, 2016
2. Pauly, Leo, and Deepa Sankar. ”Detection of drowsiness based on HOG features and SVM classifiers.” 2015 IEEE International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN). IEEE, 2015
3. Belal Alshaqaqi; Abdullah Salem Baquhaizel; Mohamed El Amine Ouis; Meriem Boumeh, 2013, “Driver drowsiness detection system”, 8th International Workshop on Systems, Signal Processing and their Applications (WoSSPA), IEEE
4. K. Satish; A. Lalitesh; K. Bhargavi; M.Sishir Prem; T Anjali, 2020, “Driver Drowsiness Detection”, International Conference on Communication and Signal Processing (ICCSP), IEEE
5. P.S. Nandhini; S. Kuppuswami; S. Malliga; P. Srinath; P. Veeramanikandan, 2022, “Driver Drowsiness Detection using Deep Learning”, 6th International
6. Aneesa Al Redhaei; Yaman Albadawi; Safia Mohamed; Ali Alnoman, 2022, “Realtime Driver Drowsiness Detection Using Machine Learning”, Advances in Science and Engineering Technology International Conferences (ASET), IEEE
7. Marwa K. Hussein; Tariq M. Salman; Abbas H. Miry; Mohammed A. Subhi, 2021, “Driver Drowsiness Detection Techniques: A Survey”, 1st Babylon International Conference on Information Technology and Science (BICITS), IEEE