A Project Report

On

DETECTING SLEEPINESS OF THE DRIVER USING IMAGE PROCESSING TECHNIQUE

Submitted to

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELGAVI, KARNATAKA- 590014

In partial fulfilment of the completion of Eighth semester

Bachelor of Engineering

In

Information Science and Engineering

By

NAMRATHA	4AL19IS026
NISHA TELLIS	4AL19IS030
SHRAVYA	4AL19IS051
VSHKER MAYENGBAM	4AL19IS063

Under the guidance of

Mr. Pradeep.V
Sr. Assistant Professor

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

ALVA'S
Education Foundation

ALVA'S INSTITUTE OF ENGINEERING AND TECHNOLOGY,
MIJAR, MOODBIDRI D.K -574225

2022-23

ALVA'S INSTITUTE OF ENGINEERING AND TECHNOLOGY

MIJAR, MOODBIDRI D.K. -574225

KARNATAKA

DEPARTMENT OF INFORMATION SCIENCE& ENGINEERING

CERTIFICATE

This is to certify that the project work entitled **"DETECTING SLEEPINESS OF THE DRIVER USING IMAGE PROCESSING TECHNIQUE"** has been successfully completed by

NAMRATHA	**4AL19IS026**
NISHA TELLIS	**4AL19IS030**
SHRAVYA	**4AL19IS051**
VSHKER MAYENGBAM	**4AL19IS063**

the bonafide students of **Alva's Institute of Engineering and Technology** in **DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING** of the **VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI** during the year 2022-23. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The Project report has been approved as it satisfies the academic requirements in respect of Project work prescribed in partial fulfillment of awarding bachelor of Engineering degree.

Mr. Pradeep V	**Dr. SUDHEER SHETTY**	**Dr. PETER FERNANDES**
Senior. Assistant Professor	**Professor**	**Principal**
Project Guide	**HOD ISE**	

Name of the Examiners **Signature with Date**

1.

2.

ALVA'S INSTITUTE OF ENGINEERING &

TECHNOLOGY MIJAR, MOODBIDRI D.K. -574225

KARNATAKA

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

DECLARATION

NAMRATHA	**4AL19IS026**
NISHA TELLIS	**4AL19IS030**
SHRAVYA	**4AL19IS051**
VSHKER MAYENGBAM	**4AL19IS063**

hereby declare that the dissertation entitled, **"DETECTING SLEEPINESS OF THE DRIVER USING IMAGE PROCESSING TECHNIQUE"** is completed and written by us under the supervision of our guide **Mr. Pradeep V, Senior Assistant Professor, Department of Information Science and Engineering, Alva's Institute of Engineering and Technology, Moodbidri**, in partial fulfillment of the requirements for the award of the degree BACHELOR OF ENGINEERING in **DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING** of the **VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI** during the academic year 2022-23.The project report is original and it has not been submitted for any other degree in any university.

NAMRATHA	4AL19IS026
NISHATELLIS	4AL19IS030
SHRAVYA	4AL19IS051
VSHKER MAYENGAM	4AL19IS063

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany a successful completion of any task would be incomplete without the mention of people who made it possible, success is the epitome of hard work and perseverance, but steadfast of all is encouraging guidance.

So, with gratitude we acknowledge all those whose guidance and encouragement served as beacon of light and crowned the effort with success.

We thank our beloved Principal **DR. PETER FERNANDES,** for his constant help and supportthroughout.

We sincerely thank, **Dr. SUDHEER SHETTY**, Professor and Head, Department of Information Science & Engineering who has been the constant driving force behind.

The selection of this Project as well as the timely completion is mainly due to the interest and persuasion of our project Guide **Mr. PRADEEP V,** Senior Assistant Professor, Department of Information Science & Engineering. We will remember his contribution for ever.

We thank our beloved Project coordinator **Mr. JAYANTKUMAR A RATHOD,** Associate Professor, Department of Information Science & Engineering for his constant help and support throughout.

We are indebted to **Management of Alva's Institute of Engineering and Technology, Mijar, Moodbidri** for providing an environment which helped us in completing our Project work.

Also, we thank all the teaching and non-teaching staff of Department of Information Science & Engineering for the help rendered.

NAMRATHA	**4AL19IS026**
NISHA TELLIS	**4AL19IS030**
SHRAVYA	**4AL19IS051**
VSHKER MAYENGBAM	**4AL19IS063**

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Drowsy driving is a serious problem that can lead to accidents and fatalities on the road. To address this issue, many researchers have explored the use of computer vision techniques to detect signs of driver drowsiness in real-time. Here , we propose a method for detecting driver drowsiness by analyzing eye closure using deep learning. Our method utilizes a camera to capture the driver's face and track the eye regions for analysis. We train the sequential model to classify eye closure patterns as indicative of drowsiness or alertness. To evaluate the effectiveness of our approach, we conduct experiments using a real-world driving dataset. Our results show that our method achieves high accuracy in detecting drowsiness and outperforms existing methods. We also discuss the limitations and future directions of our work. The proposed method has the potential to contribute to the development of safer driving technologies and reduce the number of accidents caused by drowsy driving.

INTRODUCTION

CHAPTER 1

INTRODUCTION

Drowsiness is a common problem that affects millions of people worldwide, particularly those who work long hours, drive for long periods, or suffer from sleep disorders. Drowsiness can lead to decreased alertness, slower reaction times, impaired decision-making abilities, and increased risk of accidents. The aim of this project is to provide an overview of drowsiness detection techniques and the impact of sleepiness on performance, safety, and health.

One of the most important causes of traffic accidents is driver fatigue, which is a serious threat to people's lives and property safety due to the frequency of accidents. According to theNational Highway Traffic Safety Administration (NHTSA), impaired driving caused approximately 91,000 crashes, 50,000 injuries, and 795 deaths in 2017 alone, and 72,000 crashes, 44,000 injuries, and 800 deaths in 201 in the United States. The Global Road Safety Report 2018 published by the World Health Organization (WHO) emphasizes that approximately 1.35 million people die in road accidents every year. And another 20-50 million suffer non-fatal injuries, often resulting in long-term injuries, temporary disabilities, and temporal injuries. The report also emphasizes that these deaths and injuries are largely preventable through effective road safety measures, including measures to combat driver fatigue and drowsiness.

In recent years, researchers have developed various methods for detecting drowsiness, including physiological, behavioral, and subjective measures. Three different methods have been presented to detect drowsiness, including approaches based on driver physiological signals, methods that evaluate driver performance, and methods using image-based techniques. The original category of sleep detection methods focuses on the physiological andnon-visual symptoms that appear in the body due to sleep. To achieve this, electrodes are attached to different parts of the driver's body, such as the brain, muscles, and heart, which record electrical activity. By analyzing the recorded data, the degree of sleep can be determined with reasonable accuracy. However, these methods are not ideal for practical applications because they can be distracting and uncomfortable for the driver. The category ofmethods based on the driver's performance includes as a first step the installation of sensors invarious parts of the vehicle, such as the steering wheel and accelerator pedal. Drowsiness is then detected by processing the signals received from these sensors and determining the state

of the vehicle. Although these methods have relatively high accuracy, they are not practical due to their high cost. A third group of methods uses machine vision and image processing .

technologies to analyze a sequence of video frames that record the driver's facial expressions to determine whether the driver is drowsy or on the verge of falling asleep. During drowsiness, the driver tends to almost close his eyes and sometimes yawn, which is an obvious expression of drowsiness. As such, methods based on facial feature detection are well-visualized and easy to understand. We use a real-time method based on multiple facial features to detect and warn the driver of drowsiness. The purpose of this method is to reduce the effects of light and glasses to some extent. To address this problem, researchers have explored the use of deep learning techniques such as convolutional neural networks (CNN) to detect driver drowsiness in real-time. The purpose of this project is to investigate the effectiveness of using deep learning to detect driver drowsiness. The proposed method reviews the existing literature on the subject, provides an overview of the proposed approach, and presents the results of the evaluation.

Drowsiness detection technology can significantly impact many aspects of our lives. First, it improves road safety and reduces the risk of accidents by detecting early signs of fatigue and warning drivers to take breaks. Technology can also increase productivity in the workplace by monitoring employee fatigue levels and encouraging them to take breaks and breaks as needed. Reducing errors and increasing focus improves productivity, benefiting both employees and the organization.

Second, fatigue detection technology may have beneficial effects on human health. Chronic sleep deprivation can lead to various health problems such as obesity, diabetes, and cardiovascular disease. By monitoring a person's sleep habits and alerting them to make lifestyle changes, if necessary, such technology can help improve health and reduce the risk of such diseases.

Finally, this technology can improve the quality of life of people with sleep disorders such as sleep apnea, by monitoring their sleep patterns, these individuals can be treated in a timely and It can reduce daytime sleepiness and improve cognitive function.

Drowsiness detection technology can benefit individuals, workplaces and society at large by reducing accidents, increasing productivity, improving health and enhancing quality of life

1.1 PROBLEM STATEMENT

- Existing sleep detection methods involve attaching electrodes to record electrical activity, which is uncomfortable and distracting for drivers
- Methods based on driver performance, which involve installing sensors in the vehicle, are relatively accurate but expensive. There is a need for a practical and cost-effective method for detecting drowsiness in drivers
- The proposed method uses image processing techniques to analyse the driver's face for signs of drowsiness

1.2 MOTIVATION

The motive for developing a drowsy detection system using image processing techniques is to improve driver safety and prevent accidents caused by drowsy driving. Drowsiness is a common cause of accidents, and tired drivers are more likely to make mistakes, have slower reaction times, and poor judgment. By developing an accurate and reliable real-time drowsiness detection system, we can warn drivers of danger and prevent accidents. In addition,the proposed system is non-invasive and cost-effective, making it a practical solution that can be integrated into existing vehicles to improve overall road safety.

1.3 OBJECTIVES

1. To develop an accurate and reliable system for detecting drowsiness in real-time using image processing techniques.
2. To use machine learning algorithms, such as convolutional neural networks (CNNs), to analyse facial features and detect signs of drowsiness, such as drooping eyelids, changes in facial expression, and head movements.
3. To integrate the drowsiness detection system into existing vehicles and provide real-time alerts to the driver when drowsiness is detected, using visual or auditory cues.
4. To improve the accuracy and reliability of the system by incorporating additional sensors to provide contextual information, such as vehicle speed and road conditions.
5. To evaluate the effectiveness of the system in preventing accidents caused by drowsy driving, and to compare its performance with existing methods for drowsiness detection.

1.4 ADVANTAGES

- Non-invasiveness: The image processing technique is non-invasive and does not require any electrodes or sensors to be attached to the driver's body, making it more comfortable and less distracting for the driver.

- Real-time detection: The system can detect drowsiness in real time, providing timely alerts to the driver before an accident occurs.
- Cost-effectiveness: The proposed system is cost-effective and can be easily integrated into existing vehicles, making it a practical solution for improving road safety.
- High accuracy: The system uses machine learning algorithms, such as convolutional neural networks (CNNs), to analyze facial features and detect signs of drowsiness with a high degree of accuracy and reliability.
- Flexibility: The system can be easily customized to suit different driving conditions and environments, allowing it to adapt to different driver needs and preferences.

1.5 PROJECT LIFE CYCLE

The waterfall model is a classical model used in system development life cycle to create system with a linear and sequential approach. It is termed as waterfall because the model develops systematically from one phase to another in downward fashion. The waterfall approach does not define the process to go back to the previous phase to handle changes in requirement. The waterfall approach is the earliest approach that was used for software development.

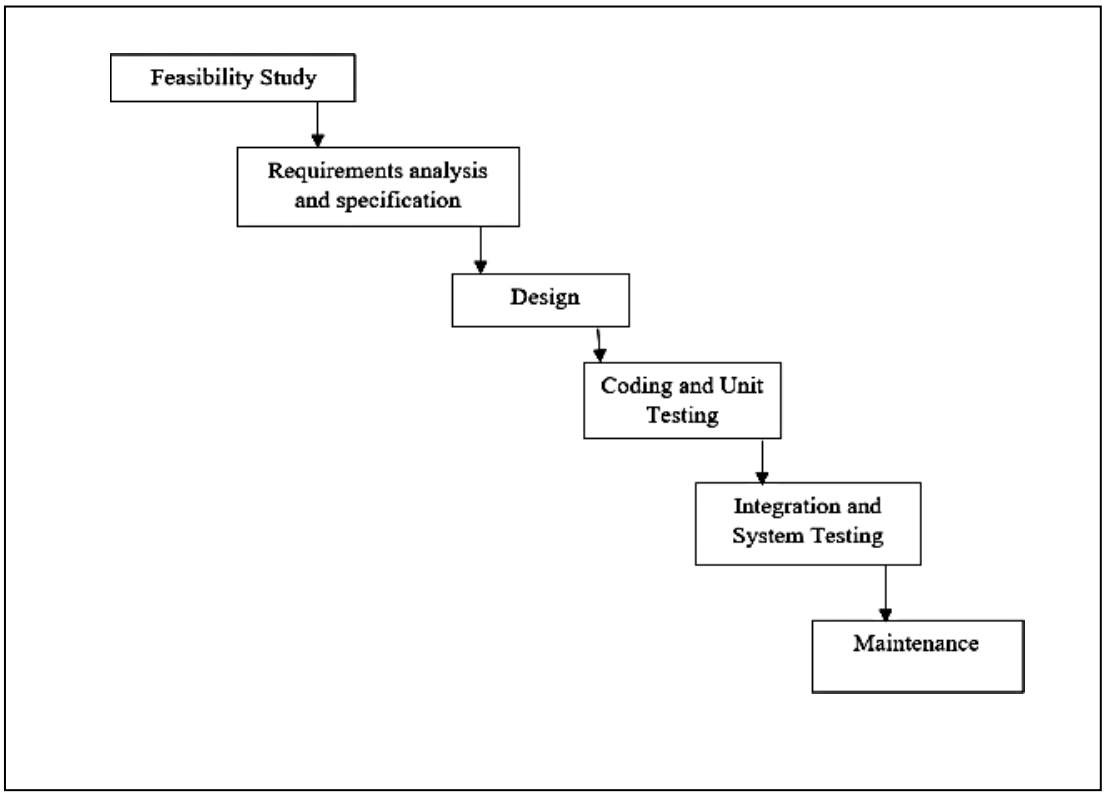

Figure 1.1 Life cycle of the waterfall model

Feasibility Study: The main goal of this phase is to determine whether it would be financially and technically feasible to develop the software.

The feasibility study involves understanding the problem and then determining the various possible strategies to solve the problem. These different identified solutions are analysed based on their benefits and drawbacks, The best solution is chosen and all the other phases are carried out as per this solution strategy.

1. Requirements analysis and specification: The aim of the requirement analysis and specification phase is to understand the exact requirements of the customer and document them properly. This phase consists of two different activities.
2. Requirement gathering and analysis: Firstly, all the requirements regarding the software are gathered from the customer and then the gathered requirements are analyzed. The goal of the analysis part is to remove incompleteness (an incomplete requirement is one in which some parts of the actual requirements have been omitted) and inconsistencies (an inconsistent requirement is one in which some part of the requirement contradicts some other part).
3. Requirement specification: These analysed requirements are documented in a software requirement specification (SRS) document. SRS document serves as a contract between the development team and customers. Any future dispute between the customers and the developers can be settled by examining the SRS document
4. Design: The goal of this phase is to convert the requirements acquired in the SRS into a format that can be coded in a programming language. It includes high-level and detailed design as well as the overall software architecture. A Software Design Document is used to document all of this effort (SDD)
5. Coding and Unit testing: In the coding phase software design is translated into source code using any suitable programming language. Thus each designed module is coded. The aim of the unit testing phase is to check whether each module is working properly or not.

1.6 INTEGRATION AND SYSTEM TESTING

Integration of different modules are undertaken soon after they have been coded and unit tested. Integration of various modules is carried out incrementally over a number of steps. During each integration step, previously planned modules are added to the partially integrated system and the resultant system is tested. Finally, after all the modules have been successfully integrated and tested.

.

System testing consists of three different kinds of testing activities as described below:

a. Alpha testing: It is the system testing performed by the development team.
b. Beta testing: Beta testing is the system testing performed by a friendly set of customers.
c. Acceptance testing: After the software has been delivered, the customer performed acceptance testing to determine whether to accept the delivered software or reject it.

1.7 MAINTANANCE

Maintenance is the most important phase of a software life cycle. The effort spent on maintenance is 60% of the total effort spent to develop a full software. There are basically three types of maintenance:

a. **Corrective Maintenance**: This type of maintenance is carried out to correct errors that were not discovered during the product development phase.
b. **Perfective Maintenance:** This type of maintenance is carried out to enhance the functionalities of the system based on the customer's request.
c. **Adaptive Maintenance**: Adaptive maintenance is usually required for porting the software to work in a new environment such as working on a new computer platform orwith a new operating system.

1.8 FUNCTIONAL REQUIREMEMTS

Real-time monitoring: The system should be able to continuously monitor the user's physiological and/or behavioral indicators to detect drowsiness in real-time, with minimal delay.

Sleep/wake state detection: The system should be able to accurately differentiate between the user's sleep and wake states, and provide timely alerts when drowsiness is detected during wakeful periods.

Alerting the driver: Once the system detects drowsiness, it must alert the driver to take corrective action. This could be done through visual, auditory, or haptic feedback, such as flashing lights, beeps, or vibrations

Maintenance and support: The system must be easy to maintain and provide technical support to ensure the system operates effectively over time.

1.9 NON-FUNCTIONAL REQUIREMENTPerformance:

The performance of the proposed system can be determined by it responsive time, time to complete the given task.

Scalability:

System should able to adopt itself to increased usage or able to handle more data as time progress.

Responsiveness:

Proposed system should be responsive to the user Input or to any external interrupt which is of highest priority and return back to same state

Useability:

User should be able to understand the flow of system easily i.e., users should able to use system without any guideline or help from experts/manuals. If user experience needs to be explained then it's not good UX.

Reliability:

The proposed system should be reliable to perform the business, i.e., when user perform some important action, it should be acknowledged with confirmation.

Security:

All the information should be secured and be encrypted with minimum needs so that it's protectedfrom outside environment also from internal attack.

1.10 APPLICATION

The system has the ability to accurately identify and classify different types of food items and estimate their calorie content. With the increasing availability of digital cameras and smartphones, food recognition and calorie measurement systems can be easily integrated into various devices and applications, making it convenient for users to monitor their food intake and nutritional requirements.

LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

2.1 STATE OF ART

The motivation of this chapter is to render the background information on the factors to be considered for this project and also its stresses to check if there is any resemblance in the workcarried out by the authors.

2.2 LITERATURE SURVEY

According to **Muhamad Dwisnanto Putro**, et al. [1], Face detection is a form of automatic face recognition that uses visual technologies. Face detection systems are crucial to the advancement of technology in face-related applications including facial emotion, verification, and identification. Robotics, artificial intelligence, the Internet of Things (IoT), additive manufacturing/3D printing, and completely autonomous vehicles are just a few of the applications for the technologies that underpin the Industrial Revolution 4.0. Typically, the face detection procedure employs video or photos as input into the computational process to count the distinguishing facial traits. Multiple layer detection on the last feature map is used to detect varied face sizes. The system result shows sequential images of face localization.

Dr Ravi Subban, et al. [2], provide improved analysis on face recognition techniques using facial feature detection techniques, taking into consideration the different variations, advantages, disadvantages and accuracy of the methods used. The recognizing the human face is simple task for a human being even seeing a person after several years. But doing the same task by a computer is not simple because the computerwill have problems in the recognizing the human face if there is a change in the facial image like, lighting conditions, complex background, pose and occlusion. According to this method, all the faces are geometrically aligned and nominalized. The main advantage of this method is the reduction in error rate and significant improvement in

face recognition. However, the method is computationally expensive. A novel approach for direct visual matching of images was using Bayesian analysis for image difference.

Savath Saypadith et al.[3] suggested a method Conventional face recognition method aims to develop a faster algorithm and more robust. Whereas, accurate recognition depends on hithe gh resolution of face image and with no occlusion. Good face image should be discriminative to the change of face identity while remaining robust to the intra- personal variation. Although face recognition algorithms have reached the accuracylevel under certain condition, the face recognition algorithm still affected by external and internal variables such as the illumination, pose, facial expression and occlusion. framework for multiple face recognition which is implemented into the embedded GPU system to recognize multiple face in real time model was trained by a network architecture that reduced the network parameter, and the tracking technique was added toa framework to reduce the processing time while keeping the acceptable recognition rate.

M. Omidyeganeh, A. Javadtalab, and S. Shirmohammadi et al**.**[4] The proposed method utilizes computer vision techniques to track the driver's face and extract the eye and mouth areas for analysis. This approach takes into account the statistical properties of the image and is more compatible with the Human Visual System (HVS) model. The eye and mouth areas are then analyzed for signs of driver fatigue, including eye closure and yawning.A camera is used in research to record face appearance. The plan is divided into four phases: extracting the face from an image, eye and mouth detection, and alert generation when asleep. These features are then merged using a fusion method toincrease the efficiency of drowsiness detection and to make more intelligent decisions. The final step is to determine the driver's state and send a warning message if drowsinessis detected.The high efficiency of the proposed idea is supported by experiments, and it has the potential to significantly reduce the number of accidents caused by driver drowsiness.Thesystem's drawback is that as a non-contact technique is employed here, itis dependent on elements such as light, cameras, and other variables.

Boon-Giin Leeet al.[5] This paper proposes a method for monitoring driver safety by

analyzing information related to fatigue using two distinct methods: eye movement monitoring and bio-signal processing. The monitoring system is designed for Android-based smartphones and receives sensory data via wireless sensor network, which is further processed to indicate the current driving aptitude of the driver. Multiple sensors are integrated and synchronized, including a video sensor to capture the driver image anda bio-signal sensor to gather the driver photoplethysmography signal. A dynamic Bayesian network framework is used for the driver fatigue evaluation, and a warning alarm is sounded if driver fatigue is believed to reach a defined threshold. However, the study reports a high false awake condition of 36%, which is not an ideal probability. Thesame case applies to other parameters such as AC, HV, and PD, which have false detection ratesof 22%, 34%, and 21%, respectively. Moreover, the true detection rate forAW is less than80% in general, with PD having the highest estimated probability of 79% and AC at 78%,while PC and HV have a low probability of around 66%. The studyconcludes that the estimated probability results are not quintessential for optimal performance. Despite this, the manifold testing of the system demonstrates the practical use of multiple features, particularly with discrete methods, and their fusion enables a more authentic and ample fatigue detection. The proposed method aims to monitor driver safety by analyzing fatigue-related information using multiple sensors and a dynamic Bayesian network framework. While the study reports some limitations in the false awake condition and truedetection rates, it demonstrates the practical use of the system's multiple features and theirfusion for a more authentic and ample fatigue.

Michelle L. Reyes et al.[6] This paper is about detecting driver distraction caused by usingin-vehicle information systems like cell phones and navigation systems. Theauthors useda data mining method called support vector machines (SVMs) to develop a real-time approach for detecting cognitive distraction using drivers' eye movements and driving performance data. They found that the SVM models were able to detect driver distraction with an average accuracy of 81.1%, outperforming traditional logistic regression models. The best-performing model used a combination of eye movement anddriving measures, summarized over a 40-second window with 95% overlap, and detecteddistraction based on experimental conditions. SVMs are suitable for measuring human

cognition as they can generate both linear and nonlinear models, extract information from noisy data, and avoid overfitting. SVM models output a binary state of 1 or 0 based on their confidence in their prediction.

Avigyan Sinha,Aneesh ,Sarada K Gopal et al.[7] This paper discusses the importance detecting drowsiness in drivers and the challenges in accurately detecting it through face and expression detection algorithms due to environmental factors such as lighting and camera positioning. The paper proposes the use of different architectures and deep learningtechniques to improve the performance of face and drowsiness detection. The system involves capturing video through a NIR camera, detecting the face using Viola Jones, DLib, or Yolo V3 algorithms, and detecting drowsiness using a modified LeNet CNN architecture. The system works by capturing video through an NIR camera, converting frames, detectingfaces, and then using deep learning to detect drowsiness. Theauthors test their proposed system and compare its performance to other detection methods. They found that their system had a high accuracy rate and was able to effectively detect drowsiness.

Gang Li, Boon Leng Lee, Wan-Young Chung, et al[8]The proposed system deals with SVMPPM, EEG headband, and smartwatch system was evaluated using a real-time driving simulation experiment with twenty participants. Fifteen subjects were used to build the model, and the remaining five subjects were used to test the model. The system achieved high accuracy rates of 91.25%, 83.78%, and 91.92% for the alert, early-warning,and full-warning groups, respectively, based on a video-based reference. The wireless EEG acquisition device includes an EEG sensory input unit and a sensory processing unit,which wirelessly transmits the digital EEG data to the smartwatch for feature extraction and probability estimation by the SVMPPM. The system can detect drowsy driving and activate a warning if the estimated probability exceeds a particular threshold. The system does not provide detailed information on the specific features used by the SVMPPM to detect drowsiness. This lack of transparency may make it difficult to understand and improve the method.

Vidhu Valsan, Paul P Mathai, Ierin Babu et al[9] The proposed system for drowsiness detection using image processing and machine learning algorithms appears to be accurate, based on the experimental results presented in the paper. The system was tested on both the Face Dataset and in real-time scenarios, and the results showed high accuracy and robustness. Additionally, the system provides a non-intrusive method of detection, as it does not require any additional sensors or equipment to be worn by the driver, which could potentially distract them from driving. the system may not perform as well in very low light conditions or if the camera quality is poor. It is important to consider these factors when evaluating the feasibility and effectiveness of the proposed system in real-world applications.

S. S. Kulkarni, A. D. Harale, and A. V. Thakur et al[10]The paper deals a real-time approach for detecting driver drowsiness due to fatigue or intoxication. The system uses image processing techniques and an embedded system with Raspbian OS to detect drowsinessby analyzing the driver's eye movement and pupil size. The system is interfaced with a microcontroller and a GSM module to issue warnings and send SMS alerts to the driver and their contacts. The paper highlights the importance of detecting driver drowsiness, which accounts for 22% of accidents, and alcohol consumption, which accounts for 33% of accidents, according to a government survey. The system uses Haar-like features to locate eyes on the face and extract features of these parts using an adaptive boost technique. The paper concludes that the system performs well for detecting driver drowsiness in real time and could potentiallyreduce the number of accidents caused by drowsiness or intoxication. The lack of clear researchquestion objectives and insufficient or biased data collection is the drawback of the work.

FUNDAMENTALS OF THE PROPOSED SYSTEM

CHAPTER 3

FUNDAMENTALS OF THE PROPOSED SYSTEM

3.1 SOFTWARE REQUIREMENTS

The software requirements are description of features and functionalities of the target system. Requirements convey the expectations convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected or unexpected from the client's point of view.

3.1.1 JUPYTER NOTEBOOK

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. It is a popular tool for data analysis, scientific computing, and machine learning tasks. Jupyter Notebook was originally developed for the Python programming language, but now supports many other programming languages such as R, Julia, and Scala. The Jupyter Notebook interface consists of a web-based interface where you can create, edit, and run Jupyter notebooks. A notebook is a document that contains executable code, formatted text, and multimedia elements. The code is executed in real-time, and the output isdisplayed in the notebook interface. Jupyter Notebook provides a powerful set of tools for data analysis, including interactive data visualization using libraries like Matplotlib, Seaborn, and Plotly. You can also use Jupyter Notebook to perform statistical analysis andmachine learning tasks using popular libraries like NumPy, Pandas, and Scikit-Learn.

Features

- A web-based interface for running code and creating interactive documents
- The ability to mix code, text, and visualizations in the same document
- Support for markdown formatting and LaTeX equations
- Integration with version control systems like Git

- Access to a large number of data science libraries and packages
- Rich Output: Jupyter Notebook provides rich output, including text, images, videos, and even interactive visualizations, making it an ideal tool for data visualization and exploration.
- Collaboration: Jupyter Notebook allows users to collaborate on documents by sharingnotebooks with colleagues or publishing them online.
- Notebooks as Documents: Jupyter Notebook allows users to create documents that combine executable code, rich text, and multimedia elements, making it an ideal toolfor teaching, research, and documentation.

System requirements

- Operating System: Windows 7 or later, macOS 10.11 or later, or a popular Linuxdistribution (e Ubuntu, Debian, Fedora, CentOS)
- Processor: 64-bit Intel or AMD processor
- Memory: at least 4 GB RAM
- Disk Space: at least 2 GB of free disk space
- Python: Python 3.3 or greater (Jupyter Notebook is compatible with Python 2.7 aswell, but it is recommended to use Python 3)
- Web Browser: Jupyter Notebook runs in a web browser, so you need a modernbrowser such as Chrome, Firefox, Safari, or Microsoft Edge.

3.1.2 PYTHON CODING LANGUAGE

Python is a high-level, interpreted programming language with a focus on code readability and simplicity. It was first released in 1991 and has since become one of the most popular programming languages in use today. Some key features of Python include:

- Dynamically typed: Unlike some other programming languages, Python does not require you to declare variable types.
- Object-oriented: Python supports object-oriented programming (OOP), allowing forthe creation and manipulation of objects and classes.
- Interpreted: Python code is executed by an interpreter, rather than being compiled

into machine code beforehand.

- Extensive standard library: Python has a large standard library that provides a wide range of tools and modules for various tasks, including web development, scientific computing, and data analysis.
- Cross-platform: Python code can be run on a variety of operating systems, including Windows, macOS, and Linux.

Python is used for following:

- Web development: Python is frequently used for building web applications and back-end services, using frameworks such as Django and Flask.
- Data analysis and visualization: Python is a popular choice for data analysis and visualization, with tools like NumPy, Pandas, and Matplotlib.
- Artificial intelligence and machine learning: Python is used extensively in AI and ML applications, with libraries such as TensorFlow, Keras, and Scikit-learn.
- Scripting and automation: Python is often used for writing scripts and automating tasks, such as file manipulation and system administration.

3.1.3 WINDOWS 10 OPERATING SYSTEM

Windows 10 is a modern operating system developed and released by Microsoft in 2015. It comes in both 32-bit and 64-bit architectures and has minimum system requirements that include a 1GHz processor, 1GB of RAM for 32-bit and 2GB of RAM for 64-bit, 16GB of free hard disk space for 32-bit and 20GB of free hard disk space for 64-bit, and a DirectX 9 or later graphics card with a WDDM 1.0 driver. Windows 10 features a new user interface called the "Universal Windows Platform" (UWP) that provides a single interface for applications that can run on any Windows device. Other new features include Cortana, a digital assistant, and Microsoft Edge, a fast and secure web browser. Windows 10 also includes several security features, including Windows Defender, which is a built-in antivirus program, Secure Boot, and BitLocker. Finally, Windows 10 is designed to

receive regular updates and new features, which are delivered through Windows Update. These features and capabilities make Windows 10 a popular choice for personal and business use.

3.1.4 OPENCV

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. It was first developed by Intel in 1999 and now is maintained by the OpenCV community.

OpenCV provides a wide range of image and video processing functionalities such as object detection, face recognition, image filtering, optical flow, stitching, and more. It supports a variety of programming languages including C++, Python, Java, and MATLAB. OpenCV also provides pre-trained models for popular computer vision tasks, such as face detection and recognition, object detection and tracking, and more.

- Here are some of the commonly used functions for video processing in OpenCV:
- Reading video frames: OpenCV provides a Video Capture class that allows you to read frames from a video file or a camera.
- Displaying video frames: You can use OpenCV's show function to display videoframes in a window.
- Writing video frames: OpenCV's Video Writer class allows you to write frames to a video file.
- Video streaming: OpenCV can stream video frames over a network using the Video Capture class and the Video Writer class.
- Video processing: You can apply various image processing techniques such as filtering, segmentation, and object detection to video frames using OpenCV's image processing functions.
- Motion detection: OpenCV provides functions for detecting motion in a video stream using background subtraction techniques.

3.1.5 NumPy

NumPy (Numerical Python) is a Python library used for scientific computing and data analysis. It provides an array object that is a multidimensional, homogeneous collection of elements

with efficient computation capabilities. NumPy arrays are more efficient than Python lists for numerical operations and are used extensively in scientific computing, machine learning, data analysis, and image processing.

Here are some of the features and functionalities of NumPy:

- Multidimensional array: NumPy provides an array object that can store data of any dimension. It can be used for 1-D, 2-D, 3-D or more dimensional data.
- Mathematical operations: NumPy provides a wide range of mathematical operations such as addition, subtraction, multiplication, and division on arrays. It also supports complexarithmetic, trigonometric, and logarithmic functions.
- Broadcasting: NumPy provides a powerful broadcasting capability that allows operationsto be performed on arrays of different shapes and sizes.
- Indexing and Slicing: NumPy allows indexing and slicing of arrays, which enables easyaccess and manipulation of the data stored in arrays.
- Array creation: NumPy provides several functions to create arrays, such as arange, linspace, and random.
- Linear Algebra: NumPy provides functions for performing linear algebra operations suchas matrix multiplication, inversion, and eigenvalue computation.
- Integration with other libraries: NumPy can be easily integrated with other Python libraries such as SciPy, Matplotlib, and Pandas.
- NumPy is widely used in image processing for image processing and analysis becauseit provides an efficient way to represent and manipulate multidimensional data such as images. Here are some ways NumPy can be used in image processing:
- Image representation: NumPy provides an efficient way to represent images as multidimensional arrays, where each element represents a pixel value. For example, a grayscale image can be represented as a 2D array, where each element represents the intensity value of a pixel.
- Image manipulation: NumPy provides functions for resizing, cropping, rotating and flipping images. These operations can be performed with arrays representing images. Image filtering: NumPy provides functions to apply various filters such as Gaussian filter, median filter and Sobel filter to images. These features can be used to remove noise, blur images, or enhance certain image features.

- Image Thresholding: NumPy provides functions to threshold an image, which is the conversion of a grayscale image to a binary image. This process can be used to segmentobjects in an image.
- Image segmentation: NumPy provides functions for image segmentation, which is the process of dividing an image into several regions based on their characteristics, such as intensity, color or texture. Image Feature Extraction: NumPy provides functions to extract features such as edges, corners and vertices from images. These features can be used to detect, identify and track an object.

3.1.6 KERAS

Keras is a popular open-source neural network library that is widely used for building and training deep learning models. It has a user-friendly API and supports both TensorFlow and Theano as backend engines, making it easy to experiment with different types of neuralnetworks. One of the key advantages of Keras is its modular design, which allows users to easily combine different types of layers and models to create complex architectures.

Keras also provides pre-trained models for a variety of applications, such as image recognition and natural language processing, and allows users to perform transfer learning to adapt pre-trained models for new tasks. The library includes visualization tools for monitoring the training process and can be easily integrated with other Python libraries suchas NumPy and Scikit-learn. Overall, Keras is a powerful and versatile deep learning librarythat is widely used in research and industry for developing cutting-edge machine learning models**.**

The key features and applications of Keras:

User-friendly API: Keras has a simple and intuitive API that allows users to quickly build and train deep neural networks. It supports both TensorFlow and Theano as backend engines.

Modular design: Keras is designed with a modular architecture that allows users to easily combine different types of layers and models to create complex neural networks.

Pre-trained models: Keras provides pre-trained models for a variety of applications,

including image recognition, natural language processing, and speech recognition.Transfer learning: Keras allows users to perform transfer learning, which involves using a pre- trained model as a starting point for a new task. This can greatly reduce the time and resources required to train a new model.Visualization tools: Keras provides built-in tools for visualizing model architectures and monitoring the training process.

Integration with other libraries: Keras can be easily integrated with other Python libraries such as NumPy, Pandas, and Scikit-learn.

Applications: Keras has been used in a wide range of applications, including image and speech recognition, natural language processing, and medical imaging.

3.2 HARDWARE REQUIREMENT

3.2.1 LAPTOP OR PC

A laptop or pc should contain the following features:

- i5 Processor
- 8GB RAM
- 100GB Hard Disk
- 2.4 GHz+ Speed

3.2.2 WEBCAM OR CAMERA

Web cameras can provide a cost-effective and convenient alternative to the proposed drowsiness detection system using image processing techniques. It can be easily integrated with most laptops and desktop computers and can record video footage in real-time. A dedicated camera can provide better image quality and accuracy, especially in low light, and can be positioned to capture the driver's face from different angles, providing more data for analysis. The choice between a webcam and a stand-alone camera depends on the specific requirements and limitations of the system and the desired performance and accuracy, the choice between a webcam and a camera will depend on the specific requirements and constraints of the system.

METHODOLOGY

CHAPTER 4

METHODOLOGY

4.1 PROPOSED SYSTEM

The proposed system for driver sleepiness detection is based on the integration of Harr cascade and Sequential model .Initially the pretrained model is trained with the custom dataset consisting of closed and open eyes, yawn and non yawning images.

The input for the system is obtained by capturing an image of the driver through the webcam. Initially, the system will detect and extract the face from the image with created Region of Interest (ROI). Next, the system will locate the eye region within the face and preprocess the detected frame and calculate eye aspect ratio. This frame will then be utilized as the input for the CNN algorithm, which will classify different levels of drowsiness. And compare the predicted value with the threshold to estimate the drowsiness.

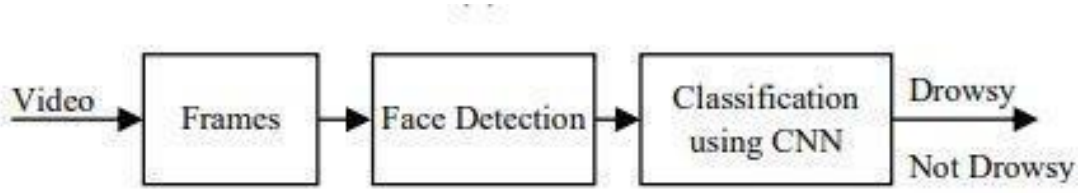

Figure 4.1: Block Diagram

SYSTEM DESIGN AND IMPLEMENTATION

CHAPTER 5

SYSTEM DESIGN AND IMPLEMENTATION

5.1 SYSTEM ARCHITECTURE

A system architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors ofthe system.

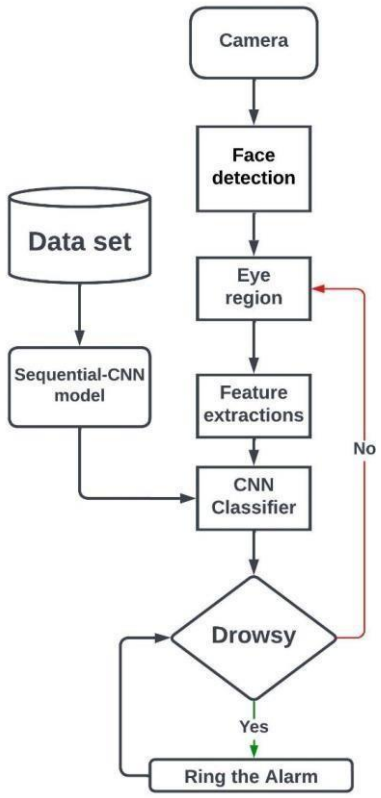

Figure 5.1: System Architecture

The above figure describes the system architecture used in the proposed system.

A. Face detection

The face detection in real time is done by Viola Jones. The Viola Jones algorithm has four

main steps.

1.Selecting Haar-like features

2.Creating an integral image

3.Running AdaBoost training

4.Creating classifier cascades

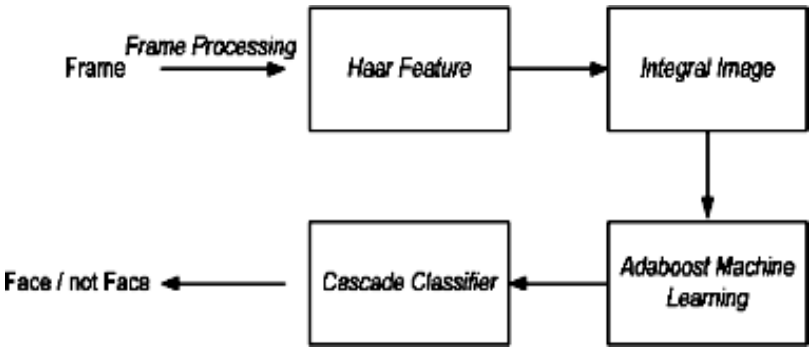

Figure 5.2 Face Detection Steps Done By Viola Jones

For detection, the method employs a Haar feature basis filter. Human faces share some characteristics, and Haar Features can be used to match them.

Haar-like features are digital image features used in object recognition. Certain characteristics of the human face are universal, such as the nose region being brighter than the eye region and the eyes region being darker than its neighbouring pixels. By subtracting the total number of pixels inside clear rectangles from the total number of pixels inside shaded rectangles, the value of every given feature may be determined.

The integral image is used to calculate the sum of pixel values in an image or a rectangular portion of an image in a quick and efficient manner.

Over 160,000 features are present in the 24x24 detector window, however only a few of these elements are significant for identifying a face. The AdaBoost method is used to find thebest features among the 160,000 features. Adaboost is short for adaptive boosting.

Each Haar-like feature is a weak classifier. The final classifier is given by a linear

combination of weak classifiers. In the AdaBoost learning algorithm, stronger classifiers are assigned larger weights.

F(x) is the strong classifier and α f(x) is the weighted combination of weak classifiers.

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) \ldots \quad (1)$$

The cascade's job is to quickly eliminate non-faces in order to avoid wasting time and calculations. We set up a cascaded system in which we divide the task of identifying a face into many steps. This is designed to remove non-faces rapidly, saving time and computational resources. Because each classifier represents a feature of a human face, a positive detection effectively states, "Yes, this subregion includes all of the characteristics of a human face." Yet, if one feature is missing, it rejects the entire subregion.

B. EYE DETECTION

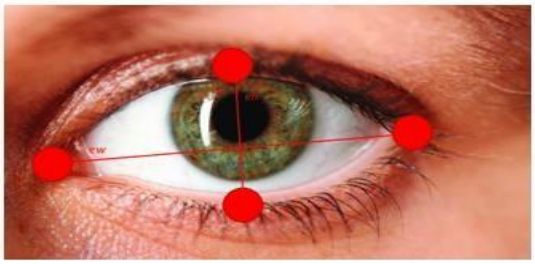

Figure 5.3 Eye Detection

The most important factor which helps detect driver fatigue is the state of eyes, i.e. open or closed.The position of the driver's eyes are determined by using Viola Jones. The system takes an image of a person's face and detects their eyes. To process each eye image, first it resizes it to a fixed size, normalizes the pixel values, and then converts it to an array. The array is then expanded to match the input shape of the neural network model, which predicts whether the eye is open or closed.

If both eyes are classified as closed, the code starts a counter and checks if the counter exceeds a certain threshold to trigger an alarm, indicating that the driver may be falling asleep.

C. CLASSIFICATION

We used the sequential model built with Keras using Convolutional Neural Networks (CNN). A convolutional neural network is a type of deep neural network that works exceptionally

well for image classification and computer vision tasks. A CNN is made up of three layers: an input layer, an output layer, and a hidden layer, which can contain several layers.

The key idea behind CNN is to learn local patterns or features from images by applying a series of convolutional filters, followed by pooling layers to extract the most important features.

The basic building blocks of a CNN are convolutional layers, pooling layers, and fully connected layers.

Convolutional layers perform a convolution operation on the input image using a set of learnable filters to extract local features or patterns. The output of a convolutional layer is a feature map.

Pooling layers reduce the spatial size of the feature maps by down-sampling the features using max pooling or average pooling. This helps in reducing the computational complexity of the network and preventing overfitting.

Fully connected layers perform a classification task on the learned features by mapping the extracted features to the output classes.

Transfer Learning

Transfer learning in convolutional neural networks (CNNs) is a technique where a pre-trained model is used as a starting point for a new task that may have different data, but similar features or classes as the original task the model was trained on. The idea is to take advantageof the knowledge that the pre-trained model has already learned during training, and then fine-tune it with the new data to adapt it to the new task.

The pre-trained model, which is typically trained on large datasets, acts as a feature extractor that maps input data to a set of high-level features. The last few layers of the pre-trained model, which are responsible for predicting the original task, are removed, and new layers areadded to perform classification or regression for the new task. These new layers are then fine-tuned on the new dataset, while the weights of the original layers are frozen or retrained witha lower learning rate.

Transfer learning can greatly reduce the amount of data and time required for training a CNN

model from scratch, and can also improve its performance, especially when the new task has limited labeled data. It is widely used in computer vision applications, such as object recognition, image classification, and image segmentation.

Sequential Model

In Keras, a Sequential model is a linear stack of layers that can be used to create a convolutional neural network (CNN) for image recognition and other computer vision tasks. It is a simple way to create a CNN architecture where layers are added sequentially, one on top of the other, like a stack.

The Sequential model in Keras is created using the Sequential() class, which can be imported from the keras.models module. The model can then be built by adding layers to it using the add() method. The Sequential model can include different types of layers, such as convolutional layers, pooling layers, and fully connected layers, that are commonly used in CNNs.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 143, 143, 256)     7168
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 71, 71, 256)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 69, 69, 128)       295040
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 34, 34, 128)       0
D)
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 64)        73792
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 64)        0
D)
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 14, 14, 32)        18464
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 7, 7, 32)          0
D)
_________________________________________________________________
flatten (Flatten)            (None, 1568)              0
_________________________________________________________________
dropout (Dropout)            (None, 1568)              0
_________________________________________________________________
dense (Dense)                (None, 64)                100416
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 260
=================================================================
Total params: 495,140
Trainable params: 495,140
Non-trainable params: 0
_________________________________________________________________
```

The input data enters the model at the top, passes through the layers, and produces the output at the bottom.

The sequential model starts with a convolutional layer (Conv2D) that applies a filter to the input image and produces a feature map. This layer has 7,168 trainable parameters. The output of the convolutional layer is passed through a max pooling layer (MaxPooling2D) that reduces the size of the feature map by taking the maximum value of each pooling window.

The second convolutional layer (Conv2D) applies another filter to the output of the first max pooling layer and produces another feature map. This layer has 295,040 trainable parameters. The output of this convolutional layer is again passed through a max pooling layer (MaxPooling2D) that reduces the size of the feature map.

The third convolutional layer (Conv2D) applies yet another filter to the output of the second max pooling layer and produces another feature map. This layer has 73,792 trainable parameters. The output of this convolutional layer is again passed through a max pooling layer (MaxPooling2D) that reduces the size of the feature map.

The fourth convolutional layer (Conv2D) applies another filter to the output of the third max pooling layer and produces another feature map. This layer has 18,464 trainable parameters. The output of this convolutional layer is again passed through a max pooling layer (MaxPooling2D) that reduces the size of the feature map.

The flattened output of the fourth max pooling layer is passed through a dropout layer (Dropout) to prevent overfitting. The output of the dropout layer is then passed through two fully connected (Dense) layers, the first with 100,416 trainable parameters and the second with 260 trainable parameters. The final output is a tensor with four values, representing the predicted probabilities for each of the four classes.

SYSTEM TESTING

CHAPTER 6

SYSTEM TESTING

Testing is an important phase in the development life cycle of the product. This is the phase where all the error remaining in all the phases will be detected. Hence testing performs a very critical role for quality assurance and ensuring the reliability of the software. During the test, the program to be tested is executed with a set of test cases and the output of the program for the tests cases is evaluated to determine whether the program is performing as expected.

Testing of software or hardware is conducted on complete system to evaluate its compliance with specified requirement. System testing is performed on entire system in the context of functional requirements specification and/or system requirement specification. Testing is an investigatory phase, where focus is to have almost a destructive attitude and test not only the design, but also the behaviour and even the believed expectation of the customer. Errors which are found are corrected by using the following testing steps and correction will be recorded for future references. Thus, a series of testing is performed on the system before it is ready for implementation.

Testing Objectives

The testing objectives are as follows:

- Testing is the process of executing the program with the intent of finding an error.
- A good test case is one that has a high probability of finding an error.
- Testing cannot show the absence of defects.

Test Types

Different types of tests are mentioned below

1. Unit testing
2. Integration testing
3. System testing

- Validation testing
- Black box testing
- White box testing

4. Acceptance testing

6.1 UNIT TESTING

A unit is the smallest piece of software that can be tested. This usually means the software can be compiled, linked or loaded into memory. A typical example in a procedural programming language would be a function/procedure or a group of these contained in a source file. In an object-oriented programming language, this typically refers to simple classes and interfaces. An example unit testing session would involve a number of calls to the unit (function, procedure or method) under test where each call might be preceded by a setup code that generates appropriate method parameters wherever required and another call performed after the test to check whether the test was successful or unsuccessful. Unit testing is the lowest testing level. It helps to refine the testing process so that reasonable system reliability can be expected when testing is performed on the next hierarchical levels. Testing at the unit level helps to expose bugs that might appear to be hidden if a big-bang testing approach was used and unit testing omitted. When these stealth bugs are not uncovered and corrected, they could cause unexpected system faults or crashes when running the system.

Unit testing is usually seen as a white-box testing technique. This is because its main focus is on the implementation of the unit being tested i.e., the class, interface, function or method under test. Unit testing seeks to find if the implementation satisfies the functional specification. It is not unusual to have system level requirements tested at the unit level for example a system might specify in its functional requirements to have a detailed complex algorithm. Since unit testing focuses on implementation and requires thorough understanding of the systems functional specification, it is usually performed by the developers. What then happens is that the programmer writes test cases after she/he has implemented the program. The obvious problem with this is that the test cases are unconsciously written to suit the programmer's implementation rather than the initial functional specification.

6.2 INTEGRATION TESTING

Integration testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing. Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems.

6.3 SYSTEM TESTING

System testing begins after completion of integration testing. System testing is done to prove that the system implementation does not meet the system requirements specification. Test planning for system testing is usually one of the first to be processed as all that is needed is the system requirements specifications and this is usually available very early in the project development lifecycle. System test planning and system testing are normally performed by a test team if there is one. System test planning phase is very dependent on the high-level design specification in the development process. As a result, any errors made in translating the requirements specification and the design specification would be very drastic as it would propagate downwards to the lower levels of test and development.

6.4 VALIDATION TESTING

Validation Testing, Validation Testing, carried out by QA professionals, is to determine if the system complies with the requirements and performs functions for which it is intended and meets the organization's goals and user needs. This kind of testing is very important, as well as verification testing. Validation is done at the end of the development process and takes place after verification is completed. Thus, to ensure customer satisfaction, developers apply validation testing. Its goal is to validate and be confident about the product or system and that it fulfils the requirements given by the customer. The acceptance of the software from the end customer is also its part. There is a notion as Independent Validation testing If the validation tests are carried out by a third party, they are known as independent validation and verification (IV&V).

The developer needs to provide the user manual to the third-party tester. This manual should clearly contain the standard working conditions of the software. These third-party organizations submit a validation report to the developer after the software is tested. The developer, upon receipt of this report, makes the required changes to the software and repeats tests it to check whether the customer needs are met or not. Software validation testing is an important part of the software development lifecycle (SDLC), apart from verification, debugging, and certification. Validation testing ensures that the software meets the quality standards set by the customer and that the product meets customer requirements.

6.5 BLACK BOX TESTING

Blackbox testing is defined as a testing technique in which functionality of the Application Under Test (AUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on software requirements and specifications. In Black Box Testing we just focus on inputs and output of the software system without bothering about internal knowledge of the software program.

Here are the generic steps followed to carry out any type of Black Box Testing

- Initially, the requirements and specifications of the system are examined.
- Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also, some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them.
- Tester determines expected outputs for all those inputs.
- Software tester constructs test cases with the selected inputs.
- The test cases are executed.
- Software tester compares the actual outputs with the expected outputs.
- Defects if any are fixed and re-tested.

6.6 WHITE BOX TESTING

White box deals with internal working of code to ensure there is no redundant code written in software. This involves testing of line of code, program, flow, logic, loop, structure, functions, class communication testing and other internal testing of program.

White box testing involves the testing of the software code for the following:

- Internal security holes
- Broken or poorly structured paths in the coding processes
- The flow of specific inputs through the code
- Expected output
- The functionality of conditional loops
- Testing of each statement, object, and function on an individual basis.

6.7 ACCEPTANCE TESTING

Acceptance testing is used to verify that the drowsiness detection system meets the requirements and specifications that were agreed upon with the stakeholders, including accuracy, sensitivity, specificity, and response time. It ensures that the system is working as intended and that it provides the necessary functionality to prevent accidents and errors.

6.8 Test Cases

TEST CASE Id	DROWSINESS DETECTION	EXPECTED OUTPUT	OBTAINED OUTPUT	RESULT
Test 1	Eyes Closed	Closed	Closed	True
Test 2	Eyes Open	Open	Open	True

Table 6.8.1 Test Cases

EXPERIMENTAL RESULTS AND ANALYSIS

CHAPTER 7

EXPERIMENTAL RESULTS AND ANALYSIS

The experimental results of our drowsiness detection system model are promising.

The model was trained for 50 epochs. During the training process, the model achieved a loss value of 11.65% and an accuracy of 95.62% on the training dataset. The validation dataset was also evaluated, and the model achieved a loss value of 08.17% and an accuracy of 96.19%.. Below graphs show the model accuracy and model loss respectively.

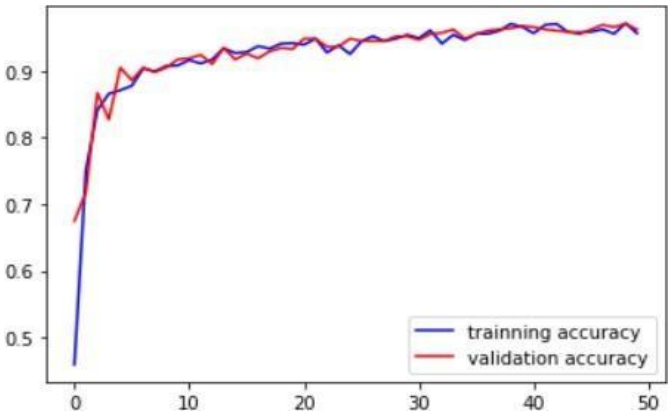

Fig 7.1 Model Accuracy

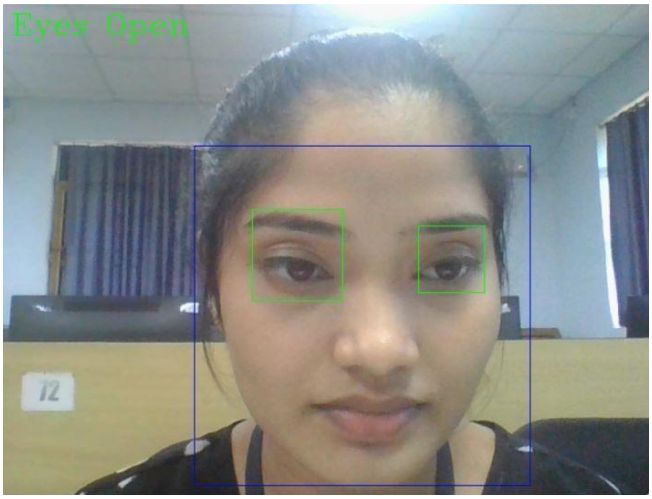

Fig 7.2 : Normal eye detection when eyes are open

Fig 7.3 : Pop out alert drowsiness notification on frame webcam when eyes are closed

CONCLUSION

CHAPTER 8

CONCLUSION

This paper proposes a drowsiness detection system based on eye detection.. The role of the system is to detect face and eyes from the video frame and extract the features of eyes which will serve as an input to the CNN model which will classify the images and predict by comparing it with the threshold and detect level of drowsiness.

According to the experimental results, it successfully detects person during drowsy condition. However, there is still space for the performance improvement.

8.1 FUTURE ENHANCEMENT

Future work could focus on collecting more diverse data that includes a wider range of individuals, lighting conditions, and driving environments to improve the accuracy and robustness of the system. Could incorporate other physiological indicators, such as heart rate variability, electroencephalography (EEG), and skin conductance, to improve the accuracy of the detection system. Could also explore the possibility of providing real-time feedback to the driver, such as adjusting the temperature or playing music to help the driver stay alert.

REFERENCES

REFERENCES

[1] M. D. Putro, Wahyono and K. -H. Jo, "Multiple Layered Deep Learning Based Real-time Face Detection," 2019 5th ICST, Yogyakarta, Indonesia, 2019, pp. 1-5, doi: 10.1109/ICST47872.2019.9166172.

[2] R. Subban and S. Soundararajan, "Human face recognition using facial feature detection techniques," 2015 International Conference on Green Computing and Internet of Things (ICGCIoT), Greater Noida, India, 2015, pp. 940-947, doi: 10.1109/ICGCIoT.2015.7380598.

[3]S. Saypadith and S. Aramvith, "Real-Time Multiple Face Recognition using Deep Learning on Embedded GPU System," 2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), Honolulu, HI, USA, 2018, pp. 1318-1324, doi: 10.23919/APSIPA.2018.8659751.

[4]M. Omidyeganeh, A. Javadtalab and S. Shirmohammadi, "Intelligent driver drowsiness detection through fusion of yawning and eye closure," 2011 IEEE International Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems Proceedings, Ottawa, ON, Canada, 2011, pp. 1-6, doi: 10.1109/VECIMS.2011.6053857.

[5]B. -G. Lee and W. -Y. Chung, "Driver Alertness Monitoring Using Fusion of Facial Features and Bio-Signals," in IEEE Sensors Journal, vol. 12, no. 7, pp. 2416-2422, July 2012, doi: 10.1109/JSEN.2012.2190505.

[6]Y. Liang, M. L. Reyes and J. D. Lee, "Real-Time Detection of Driver Cognitive Distraction Using Support Vector Machines," in IEEE Transactions on Intelligent Transportation Systems, vol. 8, no. 2, pp. 340-350, June 2007, doi: 10.1109/TITS.2007.895298.

[7]A. Sinha, R. P. Aneesh and S. K. Gopal, "Drowsiness Detection System Using Deep Learning," 2021 Seventh International Conference on Bio Signals, Images, and Instrumentation (ICBSII), Chennai, India, 2021, pp. 1-6, doi: 10.1109/ICBSII51839.2021.9445132.

[8]G. Li, B. -L. Lee and W. -Y. Chung, "Smartwatch-Based Wearable EEG System for Driver Drowsiness Detection," in IEEE Sensors Journal, vol. 15, no. 12, pp. 7169-7180, Dec.

2015, doi: 10.1109/JSEN.2015.2473679.

[9]V. Valsan A, P. P. Mathai and I. Babu, "Monitoring Driver's Drowsiness Status at Night Based on Computer Vision," 2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), Greater Noida, India, 2021, pp. 989-993, doi: 10.1109/ICCCIS51004.2021.9397180.

[10]S. S. Kulkarni, A. D. Harale and A. V. Thakur, "Image processing for driver's safety and vehicle control using Raspberry Pi and webcam," 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), Chennai, India, 2017, pp. 1288-1291, doi: 10.1109/ICPCSI.2017.8391917.