**Driver Drowsiness Monitoring System Using Deep Learning**



By

Samam-Ul-Abidin

M.Ijaz

Department of Computer Science

Govt, Akhtar Nawaz Khan (S) Degree College

KTS Haripur

Khyber Pakhtunkhwa, Pakistan

May, 2025

**Driver Drowsiness Monitoring System Using Deep Learning**



By

Samam-Ul-Abidin

M.Ijaz

Department of Computer Science

University Of Haripur

Khyber Pakhtunkhwa, Pakistan

May, 2025

**Driver Drowsiness Monitoring System Using Deep Learning**



By

Samam-Ul-Abidin, M.Ijaz

21431, 21442

Supervised by

Mr. Syed Shahid Gillani

Department of Computer Science

University Of Haripur

Khyber Pakhtunkhwa, Pakistan

May, 2025

**Driver Drowsiness Monitoring System Using Deep Learning**



By

Samam-Ul-Abidin, M.Ijaz

A Dissertation Submitted in Partial fulfillment For the Degree OF Bachelors\Master OF Science

In

Computer Science

Department of Computer Science

University Of Haripur

Khyber Pakhtunkhwa, Pakistan

May, 2025

**Driver Drowsiness Monitoring System Using Deep Learning**

By

Samam-Ul-Abidin, M.Ijaz

CERTIFICATE

A THESIS SUBMITTED IN THE PARTIAL FULFILMENT OF THE

REQUIRMENTS FOR THE DEGREE OF BACHELORS

IN COMPUTER SCIENCE

We accept this dissertation as conforming to the required standards

* Supervisor \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* External\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* Examiner\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* Chairperson\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Department of Information Technology

University of Haripur

May, 2025

Dedication

## 

This thesis is dedicated to my beloved parents and friends for their unconditional love, endless support, and constant encouragement throughout my academic journey.  
To my teachers and mentors,  
whose guidance and wisdom have shaped my path.  
And to all those who believed in me,  
even when I doubted myself.

Declaration

I hereby declare that this dissertation is neither as a whole or part thereof has been copied out from any source. If any part of this thesis is proved to be copied or found to be a report of some other, we shall bear the consequences. No portion of this work presented in the thesis has been submitted in support of any application from any other degree of qualification or any other university or any other institute of learning.

I further declare that this research and all the associated documents with this, reports and all records are submitted as a partial requirement of the degree of Bachelor in Computer Science. We understand and transfer copyrights for these materials to University of Haripur and we will not sale this research and documents for getting any financial gains. It is further declared that I developed this and also this thesis is entirely on the basis of our personal efforts made under the sincere guidance of the project supervisor Sir Syed Shahid Gillani, Department of Computer Science, University of Haripur, Haripur.

Plagiarism Certificate

This thesis has been checked for Plagiarism. Turnitin report endorsed by Supervisor is attached at the end of thesis.

Date: May 1, 2025 Student Name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Student Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Supervisor Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Acknowledgements

Allah is very kind and merciful. His benevolence and blessings enabled us to accomplish this task. We express our heartfelt gratitude to our parents and family for their prayers, moral support and sincere wishes for the completion of our work.

We would thanks to almighty Allah who gives us opportunity, the right path and help us in every problem during the project. We would like to express our sincere and humble gratitude to Almighty whose help, guidance and blessings has been a real source of all our achievements in our life.

We would thanks to our Supervisor Sir Shahid Gillani whose suggestions, encouragement and guidance is a real source of inspirations for us throughout the entire timeline of our project. Our heartfelt gratitude goes to our families for their unflagging love and supporting us spiritually throughout our life; this dissertation would have been simply impossible without them.

Abstract

These days, an ever-increasing number of professions require long time focus. Drivers should watch out for the street, so they can respond to abrupt occasions right away. Due to driving for a long time or intoxication, drivers might feel sleepy, which is the biggest distraction for them while driving. This distraction might cost the death of the driver and other passengers in the vehicle, and at the same time, it also causes the death of people in the other vehicles and pedestrians too. To prevent such accidents, we propose a system that helps to alert the driver if he/she feels drowsy. To accomplish this, we implement the solution using a computer-vision-based deep learning model. The driver’s face is detected by a face recognition algorithm continuously using a camera, and the face of the driver is captured. The face of the driver is given as input to a classification algorithm which is trained with a data set of images of drowsy and non-drowsy faces. The algorithm uses landmark detection to classify the face as drowsy or not drowsy. If the driver’s face is drowsy, a voice alert is generated by the system. This alert can make the driver aware that he/she is feeling drowsy, and the necessary actions can then be taken by the driver. This system can be used in any vehicle on the road to ensure the safety of the people who are traveling and prevent accidents that are caused due to the drowsiness of the driver.

Table of Contents

[Chapter-1 1](#_Toc197375366)

[Introduction 1](#_Toc197375367)

[1.1 Introduction 1](#_Toc197375368)

[1.2 Problem Statement 2](#_Toc197375369)

[**1.3 Nature of the problem** 3](#_Toc197375370)

[**1.3.1 High Cost of Existing Solutions** 3](#_Toc197375371)

[**1.3.2 Complex Usage Requirements** 3](#_Toc197375372)

[**1.3.3 Limited Availability** 3](#_Toc197375373)

[**1.3.4 Delayed Response Time** 3](#_Toc197375374)

[**1.3.5 Increased Workload and Maintenance** 3](#_Toc197375375)

[**1.4 Objectives of the Project** 4](#_Toc197375376)

[**1.4.1 Real-Time Drowsiness Detection** 4](#_Toc197375377)

[**1.4.2 YOLOv8 Model Integration** 4](#_Toc197375378)

[**1.4.3 Offline Application Deployment** 4](#_Toc197375379)

[**1.4.4 User-Friendly Graphical Interface** 4](#_Toc197375380)

[**1.4.5 Audible Alert System** 4](#_Toc197375381)

[**1.4.6 Platform Flexibility and Low Resource Requirement** 5](#_Toc197375382)

[**1.5 Scope of the Study** 5](#_Toc197375383)

[**1.6** **Project Organization** 5](#_Toc197375384)

[**1.7** **Outputs from the Project** 5](#_Toc197375385)

[**1.8 Organization of the Thesis** 6](#_Toc197375386)

[Chapter 2 7](#_Toc197375387)

[Existing Systems Analysis 7](#_Toc197375388)

[**2.1 Existing System** 8](#_Toc197375389)

[**2.2 Overview of Current Systems** 8](#_Toc197375390)

[**2.2.1 Camera-Based Facial Monitoring** 8](#_Toc197375391)

[**2.2.2 Vehicle-Based Behavioral Monitoring** 9](#_Toc197375392)

[**2.2.3 Wearable Sensors** 9](#_Toc197375393)

[**2.2.4 Mobile Application Solutions** 9](#_Toc197375394)

[**2.2.5 Infrared and Thermal Imaging Systems** 9](#_Toc197375395)

[**2.2.6 Hybrid Systems** 9](#_Toc197375396)

[**2.3 Limitations of Existing Systems** 10](#_Toc197375397)

[**2.3.1 Lack of Focus on Universally Accessible Solutions** 10](#_Toc197375398)

[**2.3.2 Hardware Dependency** 10](#_Toc197375399)

[**2.3.3 Limited Real-Time Responsiveness** 10](#_Toc197375400)

[**2.3.4 Inconsistent Accuracy in Natural Conditions** 10](#_Toc197375401)

[**2.3.5 Poor Accessibility and Feedback Design** 11](#_Toc197375402)

[Chapter 3 12](#_Toc197375403)

[Proposed System 12](#_Toc197375404)

[3.1. Feasibility Study 13](#_Toc197375405)

[**3.1.1 Technical Feasibility** 13](#_Toc197375406)

[**3.1.2 Economic Feasibility** 14](#_Toc197375407)

[**3.1.3 Operational Feasibility** 14](#_Toc197375408)

[**3.2. Functional Requirements** 15](#_Toc197375409)

[**3.2.1 Real-Time Image Capture** 15](#_Toc197375410)

[**3.2.2 Drowsiness Detection Model** 15](#_Toc197375411)

[**3.2.3 Text-to-Speech (TTS) Output** 15](#_Toc197375412)

[**3.2.4 Voice Command Integration** 15](#_Toc197375413)

[**3.3 Non-Functional Requirements** 16](#_Toc197375414)

[**3.3.1 High Accuracy and Precision** 16](#_Toc197375415)

[**3.3.2 Noise Tolerance** 16](#_Toc197375416)

[**3.3.3 Real-Time Performance** 16](#_Toc197375417)

[**3.3.4 Lightweight and Efficient** 16](#_Toc197375418)

[**3.3.5 User Accessibility** 16](#_Toc197375419)

[**3.4 System Requirements** 16](#_Toc197375420)

[**3.4.1 Computer/System** 17](#_Toc197375421)

[**3.4.2 Internet Availability** 17](#_Toc197375422)

[**3.4.3 GitHub Repository** 17](#_Toc197375423)

[**3.4.4 Draw.io** 17](#_Toc197375424)

[**3.4.5 Microsoft Office** 17](#_Toc197375425)

[**3.4.6 Library Resources** 17](#_Toc197375426)

[**3.4.7 Visual Studio Code** 17](#_Toc197375427)

[**3.4.8 Google Colab** 17](#_Toc197375428)

[**3.4.9 YOLOv8 Model** 18](#_Toc197375429)

[**3.4.10 Dataset** 18](#_Toc197375430)

[**3.4.11 Text-to-Speech Engine** 18](#_Toc197375431)

[**3.4.12 Basic Detection Model** 18](#_Toc197375432)

[3.5. Chosen Methodology 19](#_Toc197375433)

[**3.6. Reason** 20](#_Toc197375434)

[3.7. Use-Case Diagram 20](#_Toc197375435)

[3.8. Sequence Diagram 21](#_Toc197375436)

[3.9. DFD Diagram 21](#_Toc197375437)

[3.9. Project Timeline 22](#_Toc197375438)

[Chapter 4 23](#_Toc197375439)

[System Implementation 23](#_Toc197375440)

[4.1. Approach 24](#_Toc197375441)

[**4.1.2 Draw.io:** 24](#_Toc197375442)

[**4.1.3 Git:** 25](#_Toc197375443)

[**4.1.4 EndNote:** 25](#_Toc197375444)

[**4.1.5 Google Collab:** 26](#_Toc197375445)

[**4.1.6 Microsoft Word:** 26](#_Toc197375446)

[**4.1.7 Python:** 27](#_Toc197375447)

[4.2. Implementation 27](#_Toc197375448)

[**4.2.1 Custom Detection Model Development:** 27](#_Toc197375449)

[**4.2.2 Data Annotation and Preprocessing** 28](#_Toc197375450)

[**4.2.4 Model Training** 29](#_Toc197375451)

[4.3 GUI Development: 31](#_Toc197375452)

[4.5 Testing and Results 32](#_Toc197375453)

[Chapter 5 36](#_Toc197375454)

[Conclusion and Future Work 36](#_Toc197375455)

[**5.1. Conclusion** 37](#_Toc197375456)

[**5.2. Future Work** 37](#_Toc197375457)

[Refrences 38](#_Toc197375458)

List of Figure’s

[Figure 1 Feasibility Study 13](#_Toc197375459)

[Figure 2 Structure view of yolov8 18](#_Toc197375460)

[Figure 3 Methodology 19](#_Toc197375461)

[Figure 4 Use-case diagram 20](#_Toc197375462)

[Figure 5 Sequence Diagram 21](#_Toc197375463)

[Figure 6: DFD diagram 21](#_Toc197375464)

[Figure 7: Project timeline 22](#_Toc197375465)

[Figure 8: Vs code Environment 24](#_Toc197375466)

[Figure 9: Draw.io Environment 24](#_Toc197375467)

[Figure 10: Installation of Git Application 25](#_Toc197375468)

[Figure 11: EndNote Environment 25](#_Toc197375469)

[Figure 12: Google Collab Environment 26](#_Toc197375470)

[Figure 13: MS Word 26](#_Toc197375471)

[Figure 14: Installation of python 27](#_Toc197375472)

[Figure 15: Trained Dataset 27](#_Toc197375473)

[Figure 16: Annotating Images 28](#_Toc197375474)

[Figure 17: Preprocessing and Augmentation 29](#_Toc197375475)

[Figure 18: Download the dataset 29](#_Toc197375476)

[Figure 19: Importing libraries 30](#_Toc197375477)

[Figure 20: Data.yaml file 30](#_Toc197375478)

[Figure 21: Training on epochs 31](#_Toc197375479)

[Figure 22: Loading the model 31](#_Toc197375480)

[Figure 23: GUI interface 32](#_Toc197375481)

[Figure 24: live detection 32](#_Toc197375482)

[Figure 25: Classification output 33](#_Toc197375483)

[Figure 26: Confusion matrix 33](#_Toc197375484)

[Figure 27: Performance Evaluation 34](#_Toc197375485)

[Figure 28: Recall vs Confidence curve 34](#_Toc197375486)

[Figure 29: F1-curve 35](#_Toc197375487)

Chapter-1

Introduction

# 1.1 Introduction

According to global road safety reports and studies by the World Health Organization (WHO), driver fatigue is among the top contributors to serious road accidents (World Health Organization, 2015) [1]. Fatigue-related impairment affects over one-third of drivers worldwide, with long-haul and night-time drivers being the most vulnerable. Unlike alcohol or mobile distractions, drowsiness develops silently and is often underestimated by the driver, making it one of the most dangerous yet least addressed risk factors(Connor et al., 2001) [2].

Staying awake and alert behind the wheel is vital for safe driving. However, due to extended working hours, long-distance travel, and inadequate rest, many drivers face drowsiness, which reduces their concentration, delays their reaction time, and impairs decision-making abilities. Traditional prevention methods such as roadside signs or self-assessment tools are not sufficient to ensure timely intervention. Moreover, most existing detection systems are embedded in high-end vehicles, making them inaccessible to the average road user.

To address this, the development of real-time, affordable, and user-friendly driver monitoring systems is becoming increasingly important. Computer Vision and Artificial Intelligence (AI) offer promising solutions by analyzing facial expressions and behavioral cues to detect early signs of drowsiness. By leveraging these technologies, we can monitor driver alertness in real-time using widely available hardware like webcams.

Deep Learning, a branch of Machine Learning, has proven highly effective in pattern recognition tasks. Convolutional Neural Networks (CNNs) specifically excel in image analysis, enabling systems to detect complex features such as eye closure, yawning, and facial expressions. This project utilizes YOLOv8 (You Only Look Once version 8), a state-of-the-art deep learning model known for its accuracy and real-time processing capabilities.

The drowsiness detection process involves several core stages: live video acquisition, frame preprocessing, object detection (eyes, facial patterns), feature classification, and alert generation. These tasks are executed in a continuous loop, enabling the system to provide timely warnings.

While most advanced safety solutions rely on specialized vehicle integration, this project presents a cost-effective and portable alternative—a desktop-based driver drowsiness detection system. The application is developed in Python and uses OpenCv for real-time video processing, YOLOv8 for object detection, and Tkinter for the graphical user interface (GUI). A built-in text-to-speech engine (pyttsx3) provides audible alerts to the driver when signs of fatigue are detected.

The system works entirely offline, requiring only a standard webcam and a basic desktop or laptop. This ensures high accessibility, making the solution ideal for use in driving schools, fleet operations, or personal vehicles without built-in monitoring systems. Through this project, we aim to enhance road safety, reduce accident risks, and promote awareness of driver fatigue using a technology-driven yet accessible approach.

# 1.2 Problem Statement

Despite ongoing advancements in automobile technologies, drowsy driving continues to be a silent and underestimated risk factor on roads. Statistics reveal that a substantial percentage of traffic accidents occur due to driver fatigue, especially in commercial and long-distance transport sectors. Unlike impaired driving caused by alcohol or drugs, drowsiness lacks clear early warnings, making it extremely difficult to self-diagnose in time.

Many modern vehicles come equipped with advanced fatigue monitoring systems, but these are often limited to high-end models and require complex hardware integration such as steering sensors, lane tracking systems, or even EEG-based monitoring. These solutions are not only costly but also inaccessible for widespread use in common or older vehicles.

Furthermore, traditional methods—such as visual indicators in the dashboard or road signs reminding drivers to rest—are passive and reactive rather than proactive. They do not provide personalized, real-time feedback based on the driver’s physical condition.

Given these limitations, there is a critical need for a low-cost, accessible, and real-time drowsiness detection system that does not rely on vehicle telemetry or specialized sensors. The ideal solution should work offline, run on standard computing hardware, and use non-intrusive techniques to detect early signs of fatigue, such as frequent blinking, prolonged eye closure, or yawning.

This project addresses these challenges by introducing a desktop-based application that uses deep learning and computer vision techniques to detect driver drowsiness in real-time. By relying solely on a webcam and a trained YOLOv8 model, the system offers an affordable and scalable solution to a growing public safety issue.

## **1.3 Nature of the problem**

Based on the problem definition, the challenges associated with drowsiness detection can be categorized into the following critical areas:

### **1.3.1 High Cost of Existing Solutions**

Many existing drowsiness detection systems are integrated into high-end vehicles or rely on external hardware components such as steering sensors, seat vibration detectors, or physiological sensors. These advanced technologies significantly increase the cost of implementation and are unaffordable for low-income users and public transport fleets.

### **1.3.2 Complex Usage Requirements**

Solutions that depend on proprietary hardware or specialized vehicle integration often require professional installation and user training. This makes them less suitable for widespread adoption, especially by individuals who lack technical knowledge or resources for maintenance.

### **1.3.3 Limited Availability**

Commercial fatigue monitoring systems are typically available only in developed markets and in premium vehicle segments. Their adoption is limited in rural areas, developing countries, and among drivers using older models of vehicles, thereby restricting their potential impact on global road safety.

### **1.3.4 Delayed Response Time**

Hardware-based systems that analyze vehicular patterns (like lane deviation or steering behavior) often react after the onset of fatigue has already impaired the driver. This delayed feedback reduces the system’s effectiveness in preventing accidents.

### **1.3.5 Increased Workload and Maintenance**

Systems that rely on external sensors or embedded vehicle diagnostics require regular calibration, software updates, and technical support. They also struggle with large-scale data handling, making them inefficient for frequent and widespread use.

This project proposes a software-based, camera-only solution that mitigates these issues by offering a real-time, offline, and cost-effective alternative using basic computing resources.

## **1.4 Objectives of the Project**

The core aim of this project is to develop an intelligent and accessible driver drowsiness detection system using deep learning and computer vision technologies. The system is designed to function in real-time, operate offline, and require only minimal hardware—a standard webcam and personal computer—making it suitable for widespread adoption. The specific objectives of this project are outlined below:

### **1.4.1 Real-Time Drowsiness Detection**

To implement a robust and responsive system that continuously monitors the driver's facial features to identify signs of fatigue such as frequent blinking, prolonged eye closure, or yawning.

### **1.4.2 YOLOv8 Model Integration**

To train and deploy a YOLOv8-based deep learning model for accurate detection and classification of drowsiness-related facial expressions in live video frames.

### **1.4.3 Offline Application Deployment**

To design the system to work entirely offline without requiring internet access, ensuring reliability in remote areas and during extended travel periods.

### **1.4.4 User-Friendly Graphical Interface**

To create an intuitive and accessible GUI using Tkinter that enables users to start and stop the detection process with ease.

### **1.4.5 Audible Alert System**

To incorporate a text-to-speech (TTS) engine that provides real-time audio alerts whenever drowsiness is detected, thereby minimizing the risk of accidents.

### **1.4.6 Platform Flexibility and Low Resource Requirement**

To develop a lightweight application capable of running efficiently on standard desktops or laptops, making it accessible for a wide range of users.

These objectives collectively contribute to the development of a practical, scalable, and socially impactful solution aimed at enhancing road safety through AI-driven fatigue monitoring.

## **1.5 Scope of the Study**

This study focuses on the detection of driver drowsiness using visual indicators via a standard webcam. The system is developed in Python using OpenCv, YOLOv8, and Tkinter. It operates offline and does not rely on internet connectivity or vehicle sensors. The detection is limited to external signs of drowsiness such as eye behavior and yawning.

## **Project Organization**

**Table 1.1:** Project Organization

|  |  |
| --- | --- |
| **Members** | **Most Concerned tasks** |
| **Samam-Ul-Abidin** | **Analyzing, Model training, Validating** |
| **Muhammad Ijaz** | **Desktop-Application, Documentation** |

## **Outputs from the Project**

The completion of this project has resulted in several practical outputs that reflect both the applied technological approach and the societal relevance of the problem addressed. These outputs are categorized as follows

**1.7.1 Functional Software System**

A fully operational desktop-based application that performs real-time detection of driver drowsiness using webcam input. The software processes live video to detect facial features and behavior indicative of fatigue.

**1.7.2** **Deep Learning Integration**

Successful implementation and deployment of a YOLOv8-based object detection model, customized and trained to recognize signs of drowsiness such as eye closure and yawning.

**1.7.3 Graphical User Interface (GUI):**

A clean, accessible interface developed using Python’s Tkinter library, enabling users to easily interact with the system through basic controls.

**1.7.4 Audio Feedback Mechanism**

Integration of a text-to-speech engine that provides verbal alerts when drowsiness is detected, enhancing driver awareness without requiring visual interaction.

**1.7.5 Offline Operational Capability**

The system does not rely on internet connectivity, allowing it to function in remote or disconnected environments—a key consideration for widespread usability.

**1.7.6 Technical Documentation**

A detailed set of documents explaining the system architecture, design decisions, model training process, and deployment guidelines. These documents support future development and replication of the system.

## **1.8 Organization of the Thesis**

This thesis is structured to follow the academic guidelines and formatting standards set forth by the university. It comprises five well-defined chapters that logically present the research process and outcomes. The document begins with the preliminary pages, including the title page, acknowledgment, and abstract, followed by a detailed table of contents. Chapter 1 introduces the research problem, background, and the project's goals. Chapter 2 provides a comprehensive review of existing systems and related literature. Chapter 3 discusses the design and methodology used to develop the proposed system. Chapter 4 outlines the implementation details, results obtained, and performance evaluation. Finally, Chapter 5 concludes the study and offers recommendations for future enhancements. This structured approach ensures clarity, coherence, and thorough documentation of the project from concept to conclusion.

Chapter 2

Existing Systems Analysis

### **2.1 Existing System**

Understanding current drowsiness detection systems is crucial for identifying their strengths, weaknesses, and opportunities for improvement, which informs the design and development of our proposed solution. To gain a comprehensive understanding of the field, we employed a multi-pronged research strategy comprising the following:

Understanding the capabilities and shortcomings of current driver drowsiness detection systems is essential to identifying areas for improvement. To gain a comprehensive understanding, we employed several information-gathering methods:

• **Literature Review**: Analyzed scholarly articles, research papers, and technical reports on existing driver drowsiness detection systems and their methodologies.

• **Field Observations**: Studied real-world scenarios where drowsiness detection poses challenges, especially in long-duration driving and night shifts.

• **User Interviews**: Conducted discussions with professional drivers and individuals who frequently engage in long-distance driving to capture their experiences, challenges, and needs regarding drowsiness detection.

• **Surveys**: Distributed questionnaires to stakeholders, including drivers, transport companies, and safety experts, to evaluate the effectiveness and reliability of existing drowsiness detection systems.

This section explores the operational mechanisms of existing driver drowsiness detection systems, their strengths, and their limitations. We then assess how our proposed system, utilizing YOLOv8 for real-time driver drowsiness detection, addresses these gaps to provide a more reliable and effective solution.

### **2.2 Overview of Current Systems**

Various approaches have been developed for detecting and monitoring driver drowsiness, each employing different technologies and methodologies. The most common techniques are described below, along with their operational principles and associated drawbacks:

### **2.2.1 Camera-Based Facial Monitoring**

This method analyzes facial cues such as eye closure, blinking rate, and head posture using cameras and computer vision algorithms. Some systems use CNNs or Haar cascades to monitor these features.

* **Drawback:** Performance can be affected by lighting conditions, obstructions (like sunglasses), or camera angle misalignment.

### **2.2.2 Vehicle-Based Behavioral Monitoring**

These systems track steering behavior, lane deviation, and other vehicular parameters to estimate driver alertness. They are commonly integrated into luxury vehicles.

* **Drawback:** Reactive in nature, these systems detect symptoms after the driver is already impaired. They also rely on vehicle integration, limiting general use.

### **2.2.3 Wearable Sensors**

Wearable devices like EEG headbands, smartwatches, and specialized glasses measure physiological signals such as brain activity, heart rate, and eye movement.

* **Drawback:** Though highly accurate, these solutions are often expensive, uncomfortable for long-term use, and require regular maintenance.

### **2.2.4 Mobile Application Solutions**

Some smartphone apps utilize the front camera to monitor driver alertness or track head movement while mounted on the dashboard.

* **Drawback:** Effectiveness depends on stable camera positioning, battery power, and ambient lighting. These apps also require constant user setup and are prone to distractions.

### **2.2.5 Infrared and Thermal Imaging Systems**

These advanced systems use infrared sensors to monitor eye and facial movements in low-light conditions.

* **Drawback:** Cost-intensive and not widely accessible. Requires complex hardware and specialized installation.

### **2.2.6 Hybrid Systems**

A combination of vehicle behavior analysis and facial recognition for improved accuracy and reliability.

* **Drawback:** High implementation complexity and cost; often reserved for research or premium applications.

These existing methods highlight the diverse yet fragmented nature of current fatigue monitoring solutions. Most require proprietary hardware or are limited by environment-specific conditions. In contrast, our proposed system leverages an open-source deep learning model (YOLOv8), standard webcam, and desktop GUI—designed to be affordable, accessible, and effective in real-world scenarios without the need for specialized equipment.

### **2.3 Limitations of Existing Systems**

Despite the technological diversity in current drowsiness detection methods, several critical limitations restrict their effectiveness, especially for mass adoption in low-resource or everyday environments. These limitations justify the need for a simplified, accessible, and more inclusive alternative, as outlined below:

### **2.3.1 Lack of Focus on Universally Accessible Solutions**

Most existing systems are tailored for specific markets or integrated into high-end vehicles, neglecting widespread usability in personal or older vehicles.

### **2.3.2 Hardware Dependency**

Many commercial and research systems rely on specialized hardware [3] such as, EEG sensors, or vehicle-integrated modules, making them cost-prohibitive and inaccessible for everyday drivers.

### **2.3.3 Limited Real-Time Responsiveness**

Some methods, especially those based on physiological sensors or complex model architectures, suffer from delays or require calibration time, making them less effective in urgent or fast-paced scenarios.

### **2.3.4 Inconsistent Accuracy in Natural Conditions**

Environmental factors such as lighting changes, face obstructions, or head movement can disrupt camera-based systems, leading to unreliable detection in uncontrolled conditions.

### **2.3.5 Poor Accessibility and Feedback Design**

Existing systems often provide visual alerts or dashboard notifications, which may not be adequate for all drivers. Few systems incorporate multimodal alerts (e.g., voice `warnings), limiting accessibility and responsiveness.

By identifying these limitations, this section underscores the necessity for a lightweight, real-time, and non-intrusive solution—precisely the gap our proposed system intends to fill.. Most require proprietary hardware or are limited by environment-specific conditions. In contrast, our proposed system leverages an open-source deep learning model (YOLOv8), standard webcam, and desktop GUI—designed to be affordable, accessible, and effective in real-world scenarios without the need for specialized equipment.

Chapter 3

Proposed System

# 

# 3.1. Feasibility Study

To tackle the serious issue of fatigue-induced road accidents, a sophisticated driver drowsiness detection system was conceptualized and built using Python. This intelligent application harnesses the power of computer vision to continuously monitor and analyze live video feed from a webcam, identifying early signs of fatigue such as prolonged eye closure and yawning. At its core, the system combines the precision of YOLOv8 for detection, the flexibility of OpenCV for image processing, and the simplicity of Tkinter to craft an intuitive user interface—making the entire solution both powerful and approachable [5][6].

Before implementing a system, it is essential to assess its feasibility from technical, economic, and operational perspectives. To ensure the practicality and effectiveness of the proposed driver drowsiness detection system, a comprehensive feasibility analysis was conducted. The following section outlines the technical feasibility that underpins this project:

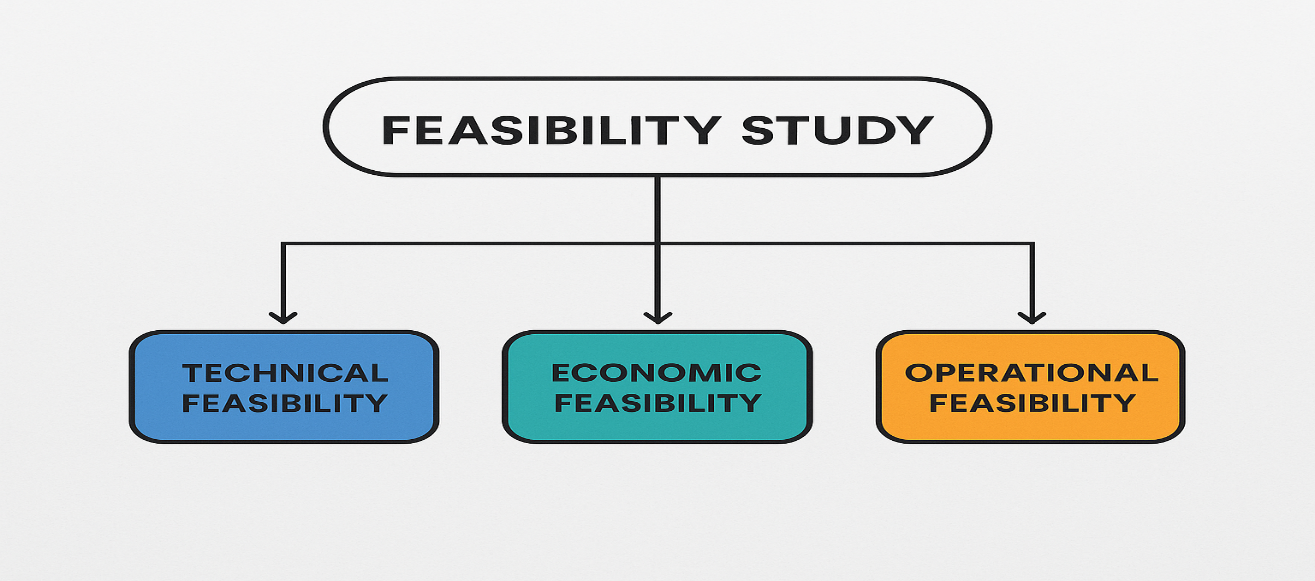


Figure 1 Feasibility Study

### **3.1.1 Technical Feasibility**

The technical feasibility assessment evaluates whether the technologies, tools, and resources required for this system are readily available and capable of supporting its development and deployment.

* The system is developed using **Python** as the core programming language, selected for its strong support for AI, deep learning, and GUI development.
* **OpenCV** is employed for real-time video capture and image preprocessing, enabling efficient handling of visual input from a standard webcam.
* The core detection model is based on **YOLOv8**, a state-of-the-art deep learning algorithm known for high-speed, accurate object detection. This model is trained to recognize facial cues such as eye closure and yawning (Jocher et al., 2023) [4].
* **Tkinter** is used for building a lightweight, desktop-friendly graphical user interface, making the system easy to operate.
* Audio feedback is delivered through **pyttsx3**, a text-to-speech engine, which provides immediate alerts when drowsiness is detected.
* Model training and testing were conducted using **Google Colab**, a cloud-based development platform that provides high-performance GPU access. This environment was particularly useful for processing large datasets and accelerating the training phase without relying on local computational resources.

### **3.1.2 Economic Feasibility**

The economic feasibility evaluates the cost-effectiveness of implementing and maintaining the proposed system. This project was intentionally designed with affordability and scalability in mind:

* Most of the software tools and libraries used in this system—such as **Python**, **OpenCV**, **Tkinter**, **PyTorch**, and **pyttsx3**—are **open-source and freely available**, eliminating the need for costly licenses.
* The only minor expense incurred is a subscription to **Google Colab Pro**, which enhances training efficiency by offering access to faster GPUs and removing session time limits. This minimal cost is justified by the significant improvements it brings in training speed and convenience [7].
* Since the application is built to run on a Raspberry Pi with a connected camera module or commonly available webcams, it offers an affordable and easily deployable solution in both personal and professional settings without the need for specialized or high-end hardware.

### **3.1.3 Operational Feasibility**

The operational feasibility determines how effectively the proposed system can be integrated into its intended environment and whether it addresses the real-world problems it is designed to solve.

* The driver drowsiness detection system is designed to enhance road safety by actively monitoring signs of fatigue in real time, thus addressing one of the leading causes of traffic accidents.
* The system provides audio alerts through a text-to-speech engine, allowing drivers to be notified without taking their eyes off the road or engaging with a visual interface.
* With a simple interface and minimal user interaction required, the system is practical and convenient for everyday use. It requires only a standard webcam and desktop or laptop, ensuring ease of adoption.
* The application is suitable for a variety of environments, including private vehicles, training centers, and fleet management operations, without necessitating technical expertise or complex setup procedures.

Overall, the system’s design emphasizes simplicity, efficiency, and accessibility, ensuring it can operate effectively in real-world settings to promote driver safety.

## **3.2. Functional Requirements**

These are the core features and behaviors the system **must perform** to meet the needs of visually impaired users for real-time currency detection:

### **3.2.1 Real-Time Image Capture**

The system must continuously capture frames from the connected webcam to enable real-time monitoring of the driver’s facial expressions and behavior.

### **3.2.2 Drowsiness Detection Model**

The YOLOv8-based model should process each captured frame to detect visual cues such as closed eyes, yawning, or head tilting with high accuracy using the trained dataset.

### **3.2.3 Text-to-Speech (TTS) Output**

Upon detecting signs of drowsiness, the system must immediately provide an audible alert using the integrated text-to-speech engine to notify the driver.

### **3.2.4 Voice Command Integration**

The system should support basic voice commands (e.g., “start detection”) to improve usability, especially during driving, and allow hands-free control of the application.

These features ensure that the system performs its core task effectively while offering a user-friendly and accessible interface suitable for a wide range of users.

## **3.3 Non-Functional Requirements**

The non-functional requirements define the system’s performance characteristics and constraints to ensure a smooth, reliable, and accessible user experience:

### **3.3.1 High Accuracy and Precision**

The detection model must maintain high precision and recall rates to minimize false positives and false negatives during drowsiness detection.

### **3.3.2 Noise Tolerance**

The system should operate reliably even under varying lighting conditions, facial orientations, and minor occlusions, such as eyeglasses or partial face visibility.

### **3.3.3 Real-Time Performance**

Detection and audio feedback must occur with minimal delay, ensuring alerts are issued promptly to effectively intervene in potentially hazardous situations.

### **3.3.4 Lightweight and Efficient**

The application must be optimized for use on standard desktops or laptops without requiring advanced graphics hardware, ensuring smooth operation and broad accessibility.

### **3.3.5 User Accessibility**

The user interface should be minimalistic and intuitive, supporting easy interaction and designed with inclusivity in mind—particularly for users who may benefit from hands-free or auditory interaction.

## **3.4 System Requirements**

To ensure the successful development and execution of the driver drowsiness detection system, several software and hardware resources are required. These include programming environments, libraries, tools for modeling and documentation, and supportive infrastructure. The requirements are categorized below:

### **3.4.1 Computer/System**

A Raspberry Pi with sufficient processing power, along with a connected camera module, is required to support real-time video processing and deep learning inference.

### **3.4.2 Internet Availability**

An active internet connection is necessary for downloading dependencies, accessing development resources, and collaborating via cloud platforms like Google Colab.

### **3.4.3 GitHub Repository**

Used for version control, code storage, sharing datasets, and collaborating with peers during the development phase.

### **3.4.4 Draw.io**

Helpful for designing system diagrams such as architecture layouts, flowcharts, and data pipelines to visualize system functionality.

### **3.4.5 Microsoft Office**

Used for documenting the thesis in MS Word, generating performance reports in MS Excel, and creating final presentations in MS PowerPoint.

### **3.4.6 Library Resources**

Academic journals, books, and thesis collections used for understanding drowsiness detection research and referencing previous works.

### **3.4.7 Visual Studio Code**

The primary integrated development environment (IDE) used for writing and debugging the Python code for both model integration and GUI design.

### **3.4.8 Google Colab**

Used as a cloud-based development platform with GPU support to train the YOLOv8 model on custom datasets efficiently.

### **3.4.9 YOLOv8 Model**

The deep learning model used for object detection and drowsiness classification. It is trained to recognize facial features such as closed eyes and yawns.

### **3.4.10 Dataset**

Custom dataset consisting of images annotated to include drowsy, yawning, and sleeping facial states. It serves as the training foundation for the YOLOv8 model.

### **3.4.11 Text-to-Speech Engine**

A TTS engine (e.g., pyttsx3) is used to deliver audio alerts to the driver when drowsiness is detected, enhancing user engagement and safety.

### **3.4.12 Basic Detection Model**

The trained YOLOv8 model with pre=trained weights is fine-tuned using the collected dataset to create an effective drowsiness detection solution. A visual diagram of the YOLOv8 pipeline can be added to demonstrate its operational flow.

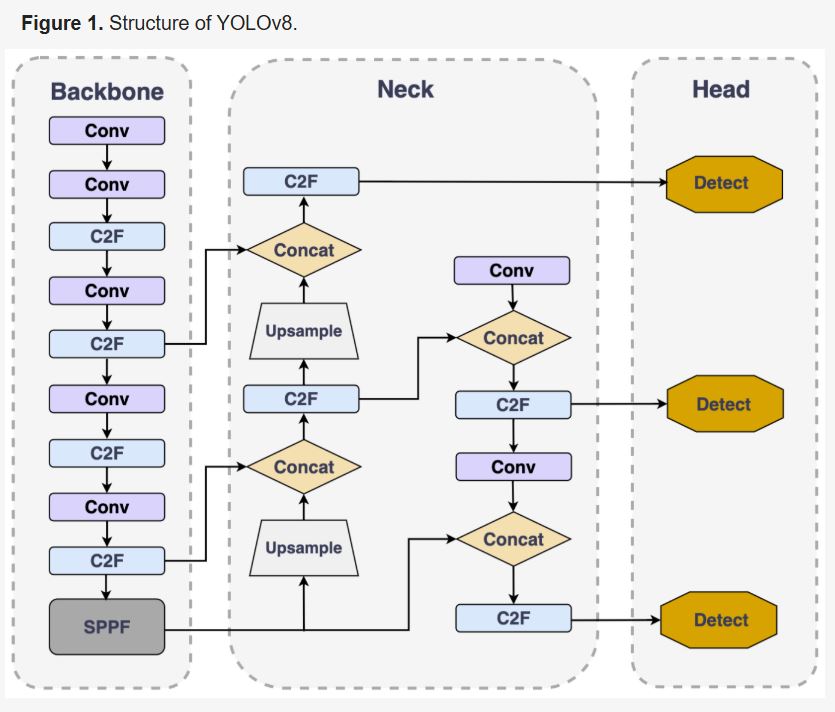


Figure 2 Structure view of yolov8

*Since the YOLO model was first introduced, the series has undergone multiple updates and iterations, with continually enhanced performance. As the most recent development in the YOLO model series, YOLOv8 represents the current pinnacle of technology. Its architecture, depicted in*[***Figure 1***](https://www.mdpi.com/1424-8220/24/15/4858#fig_body_display_sensors-24-04858-f001)*, comprises three main components: the backbone, the detection head, and the neck*

The architecture consists of the following main components:

1. **Backbone:** Responsible for extracting essential visual features from the input image.
2. **Neck:** Constructs feature pyramids that help detect objects at multiple scales and sizes, improving accuracy across varied input conditions.
3. **Head:** Final detection stage that predicts bounding boxes, object scores, and class labels based on the extracted features.

This streamlined architecture allows the model to detect fatigue symptoms from webcam feeds quickly and efficiently, making it well-suited for the requirements of a real-time driver monitoring system.

# 3.5. Chosen Methodology



Figure 3 Methodology

The **Agile methodology** is ideal for software projects that involve continuous improvement, experimentation, and adaptation—perfectly aligning with how you developed your project in stages like model training, GUI integration, and real-time testing.

### **3.6. Reason**

We selected the agile methodology for this project because it promotes flexible, iterative development and is well-suited to projects involving real-time testing and continuous improvement. The development of the driver drowsiness detection system required multiple testing cycles, refinements of the YOLOv8 model, and ongoing adjustments to the GUI and alert system. Agile’s sprint-based structure allowed us to break the project into manageable phases, prioritize important features, and improve the system based on feedback from each cycle. This helped us ensure that the system was not only technically effective but also practical and user-friendly in real-world driving conditions [8].

# 3.7. Use-Case Diagram

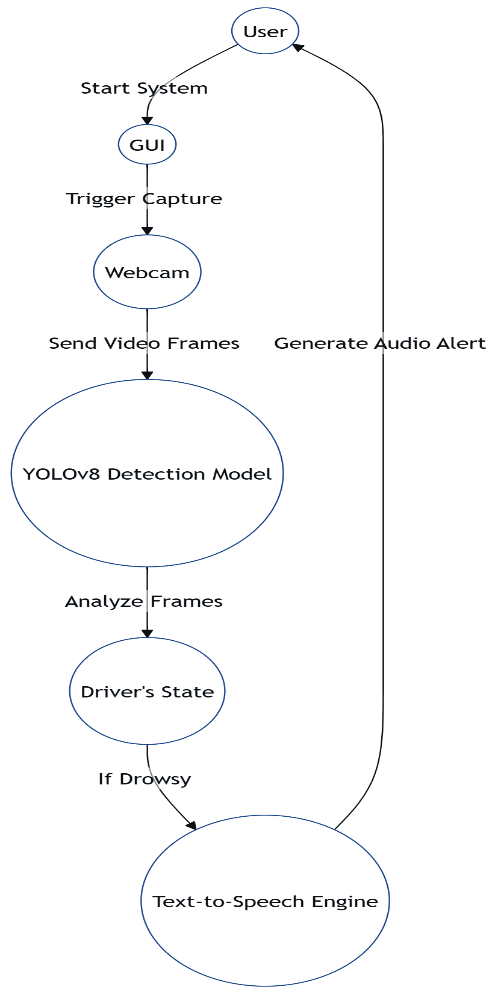


Figure 4 Use-case diagram

# 3.8. Sequence Diagram

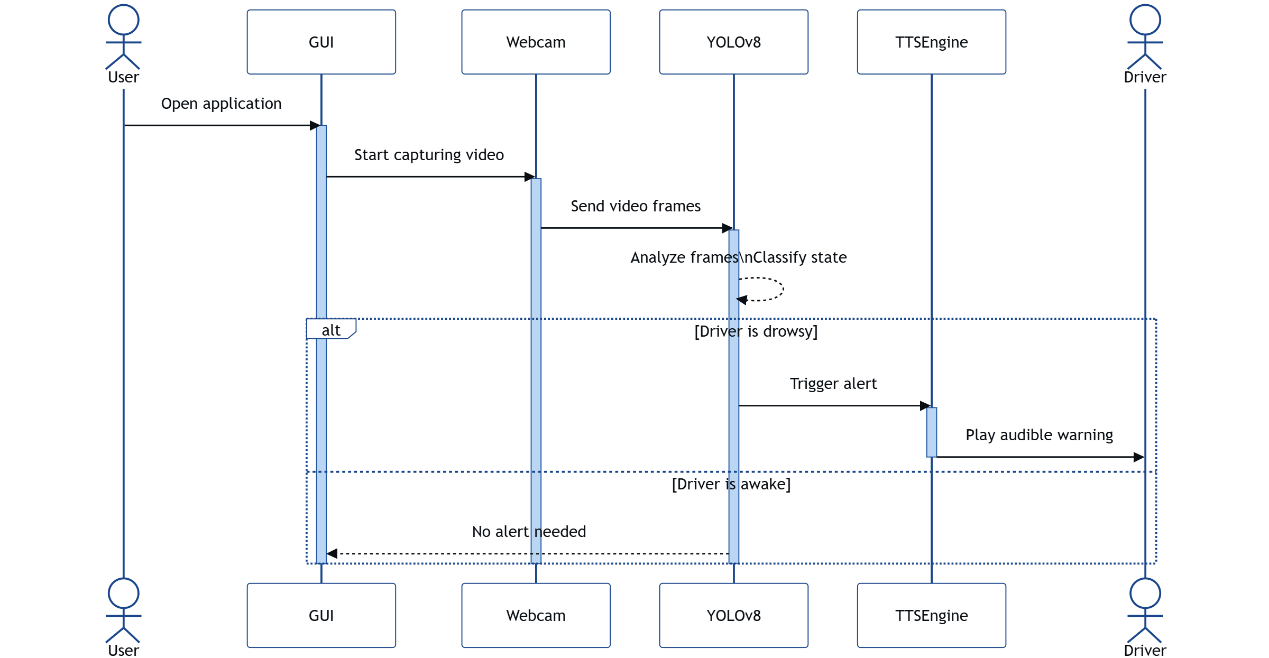
****

Figure 5 Sequence Diagram

# 3.9. DFD Diagram

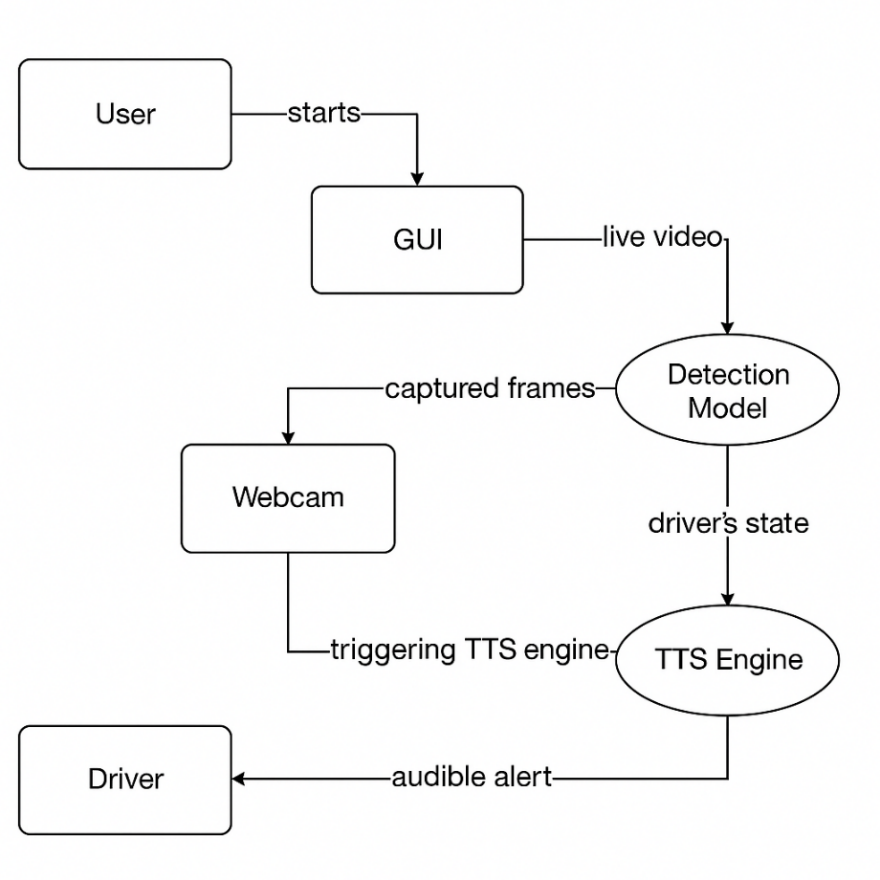


Figure 6: DFD diagram

# 3.9. Project Timeline

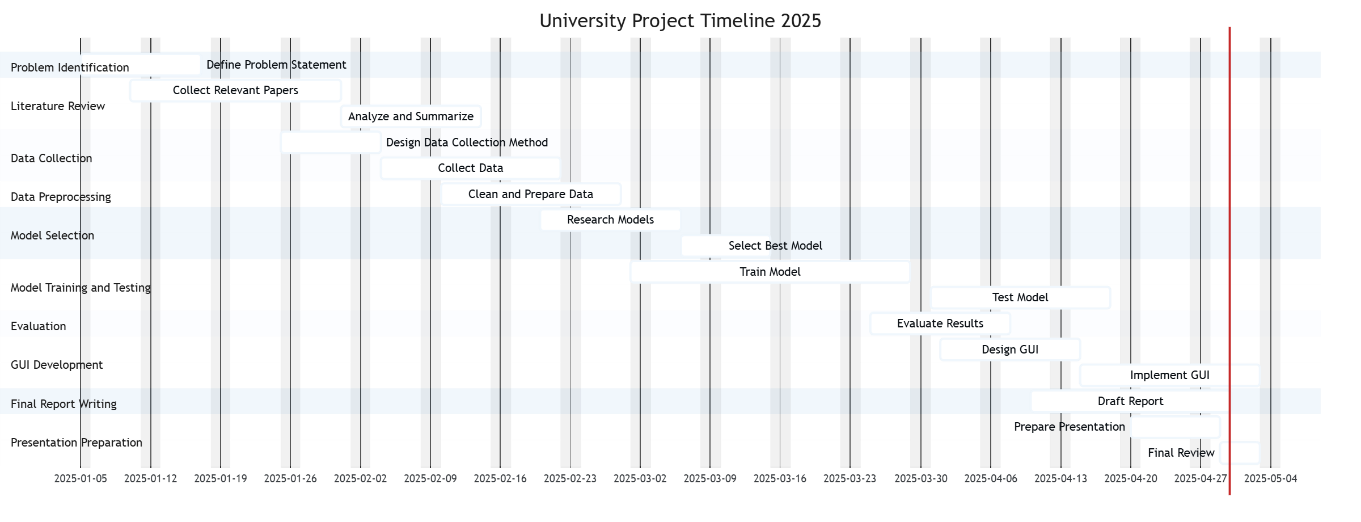


Figure 7: Project timeline

Chapter 4

System Implementation

# 4.1. Approach

The approach is that we need to install all the dependencies of our project.

**4.1.1Visual-Studio:**

Visual Studio Code will serve as our Integrated Development Environment (IDE) for developing the Desktop application (GUI).



Figure 8: Vs code Environment

### **4.1.2 Draw.io:**

For making of different diagram of our project, we use the online tool draw.io

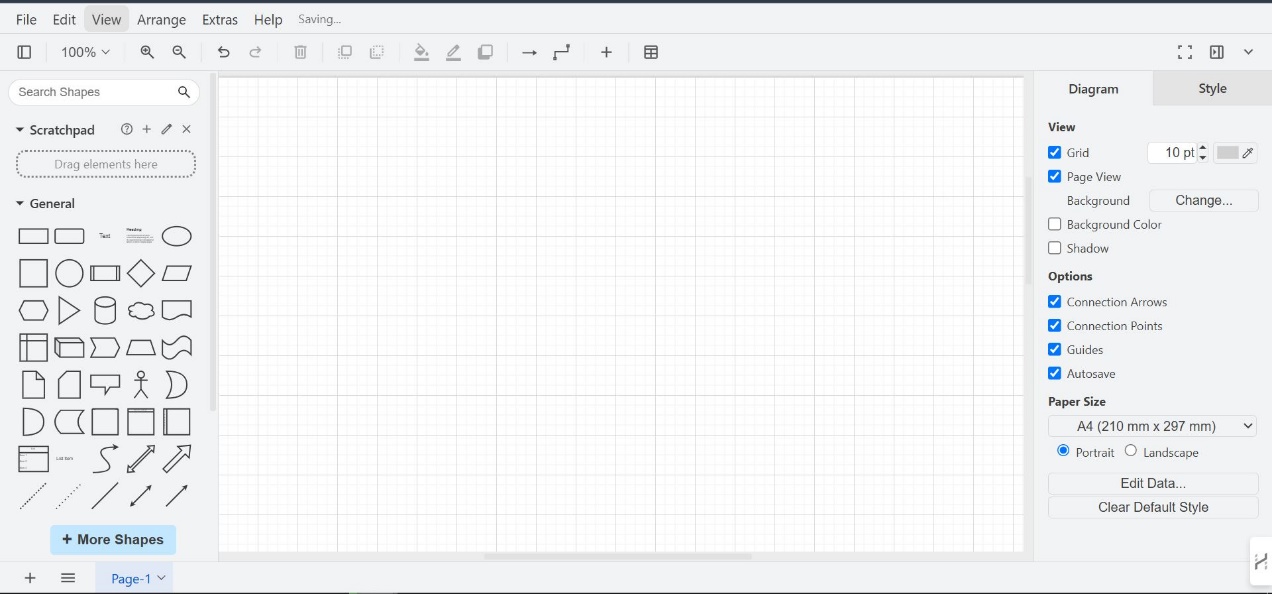


Figure 9: Draw.io Environment

### **4.1.3 Git:**

To cloan the yolov8 model reposaratory we need Git Application from GitHub.

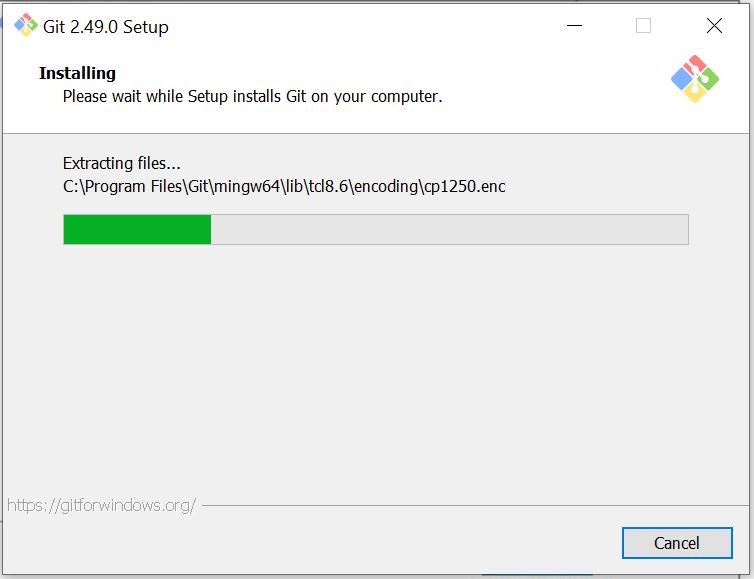


Figure 10: Installation of Git Application

### **4.1.4 EndNote:**

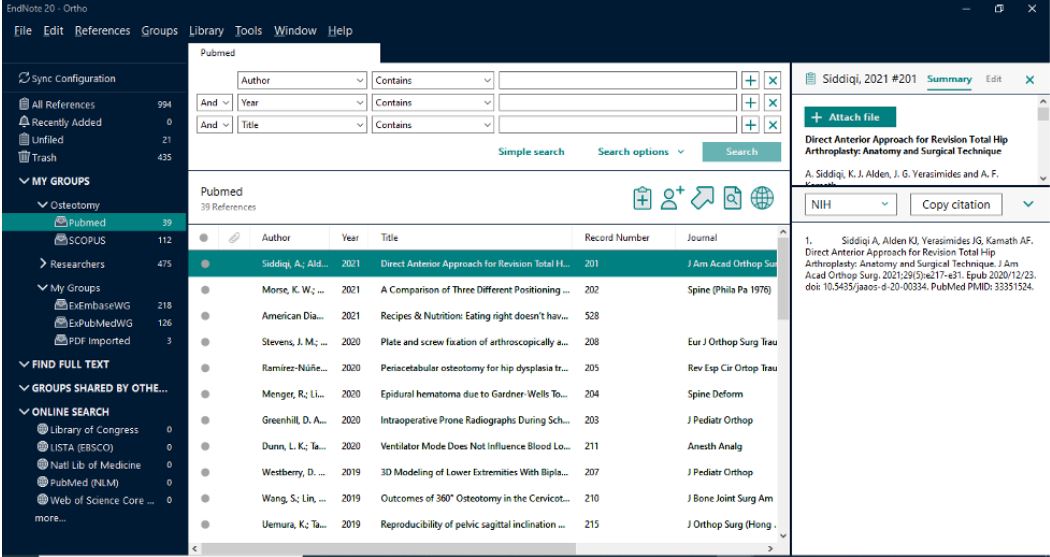
For managing the references during the documentation of our project we use the EndNote.

Figure 11: EndNote Environment

### **4.1.5 Google Collab:**

The google colab is a free, cloud-based platform that allows you to write and execute the Python code in browser. We use it for our model tanning and validation etc.

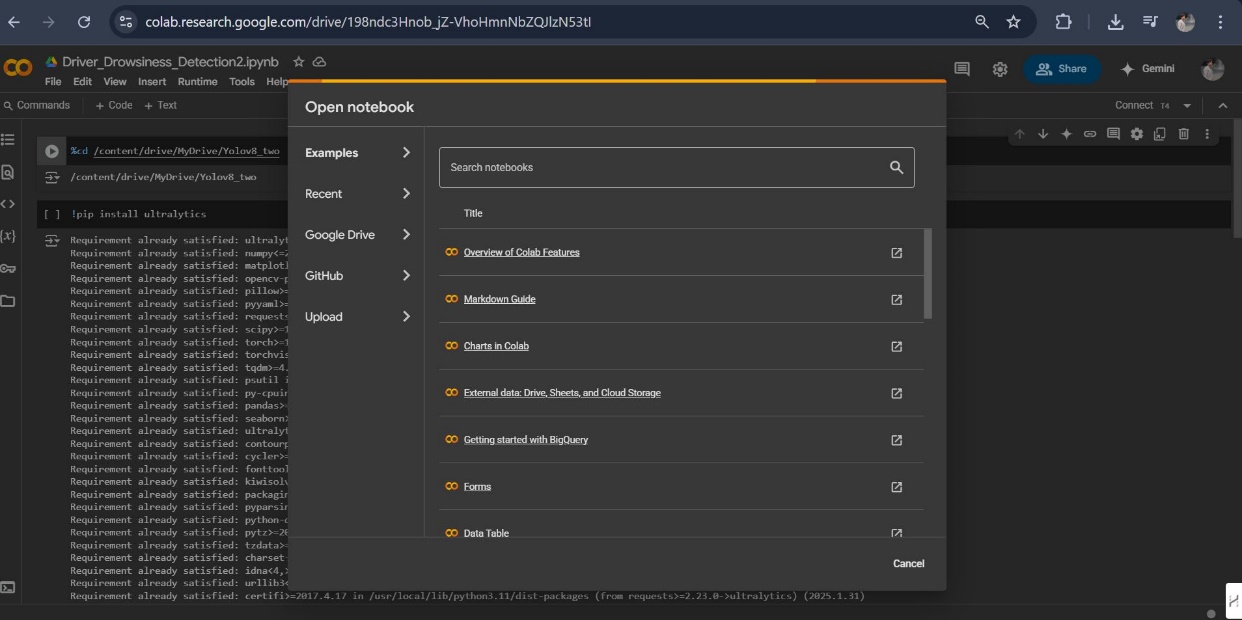


Figure 12: Google Collab Environment

### **4.1.6 Microsoft Word:**

We will utilize Microsoft Word to record every stage of our project, ensuring consistency by following a standardized documentation format.

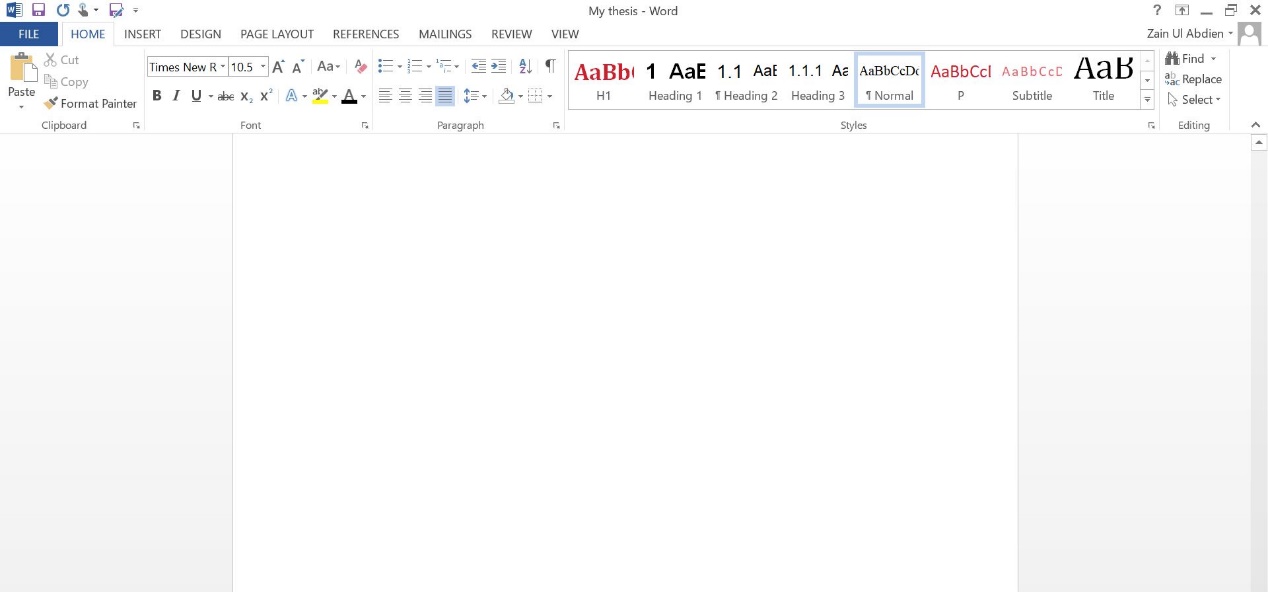


Figure 13: MS Word

### **4.1.7 Python:**

It is necessary to install the Python interpreter to enable the use of essential libraries in our code.

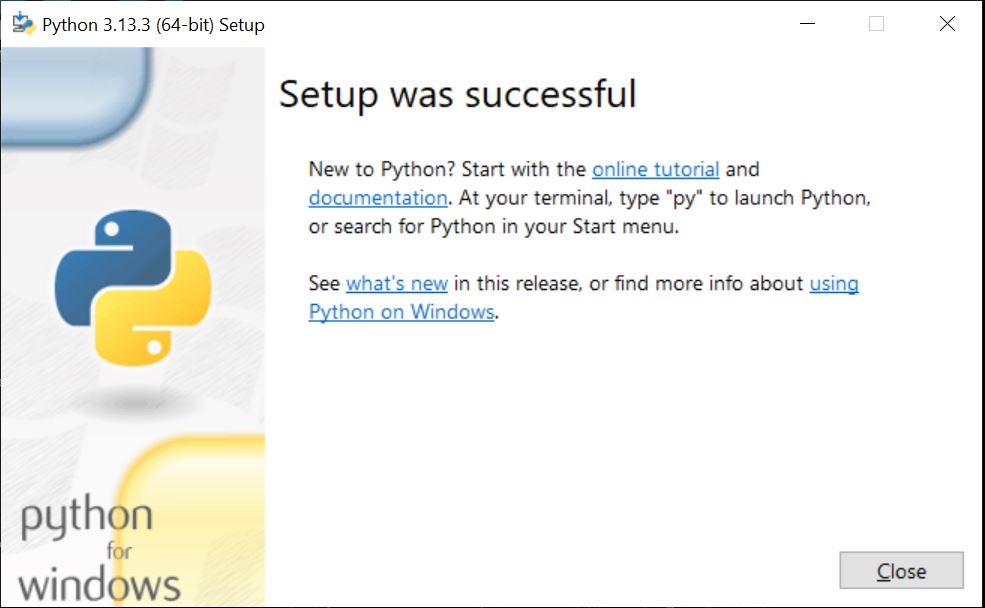


Figure 14: Installation of python

# 4.2. Implementation

### **4.2.1 Custom Detection Model Development:**

With all the prerequisites in place, we can now begin developing our detection model. The first step in this process is to create the dataset.

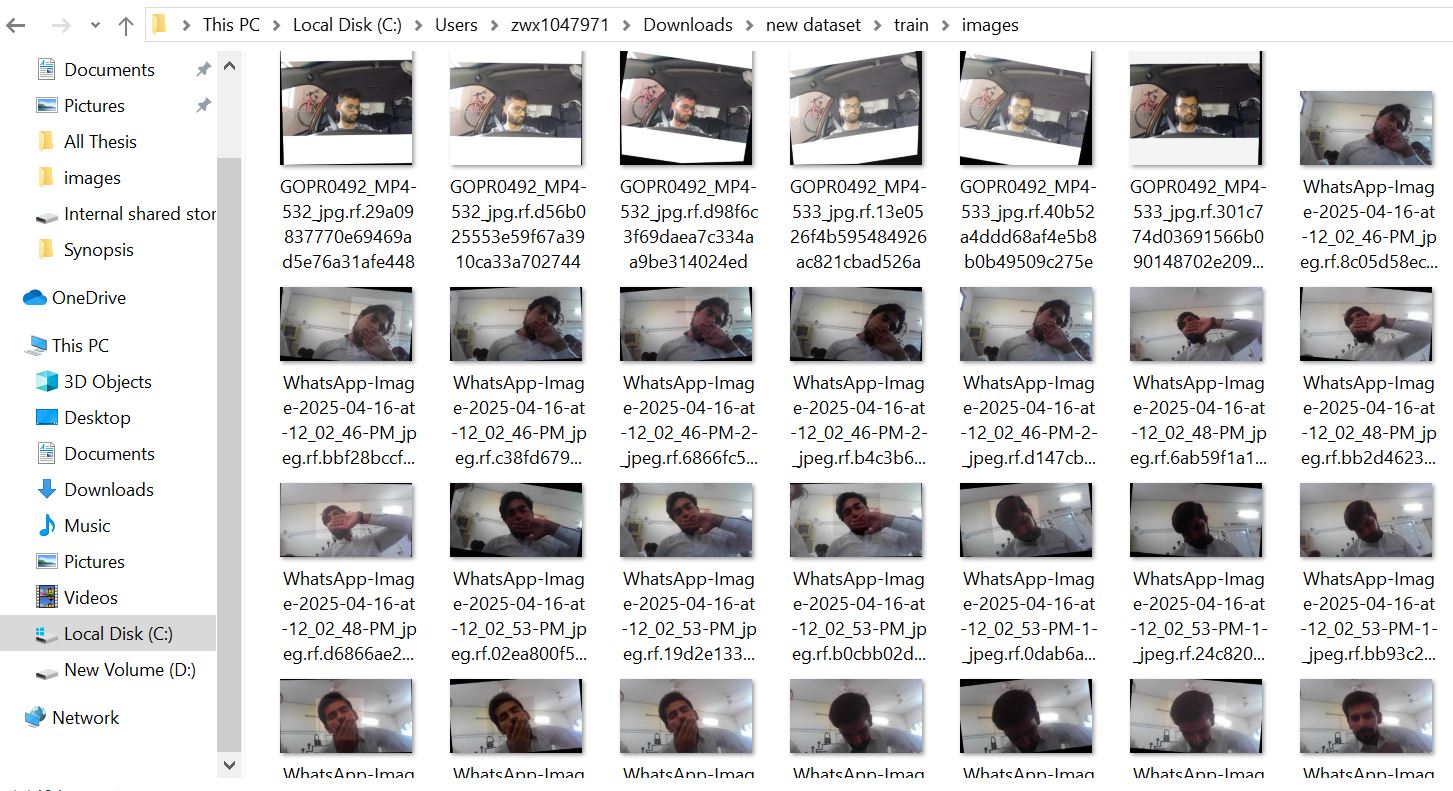


Figure 15: Trained Dataset

### **4.2.2 Data Annotation and Preprocessing**

To prepare our dataset for training the YOLOv8 detection model, we used **Roboflow**, a popular online platform for image annotation and dataset preprocessing. Roboflow allows users to easily upload, annotate, and export datasets in formats compatible with YOLO models.

* **Annotate facial features** like closed eyes and yawning by drawing bounding boxes on relevant regions of each image.
* Label **each instance** with class names (e.g., "drowsy", "yawning", "sleep") to help the model learn to classify different states.

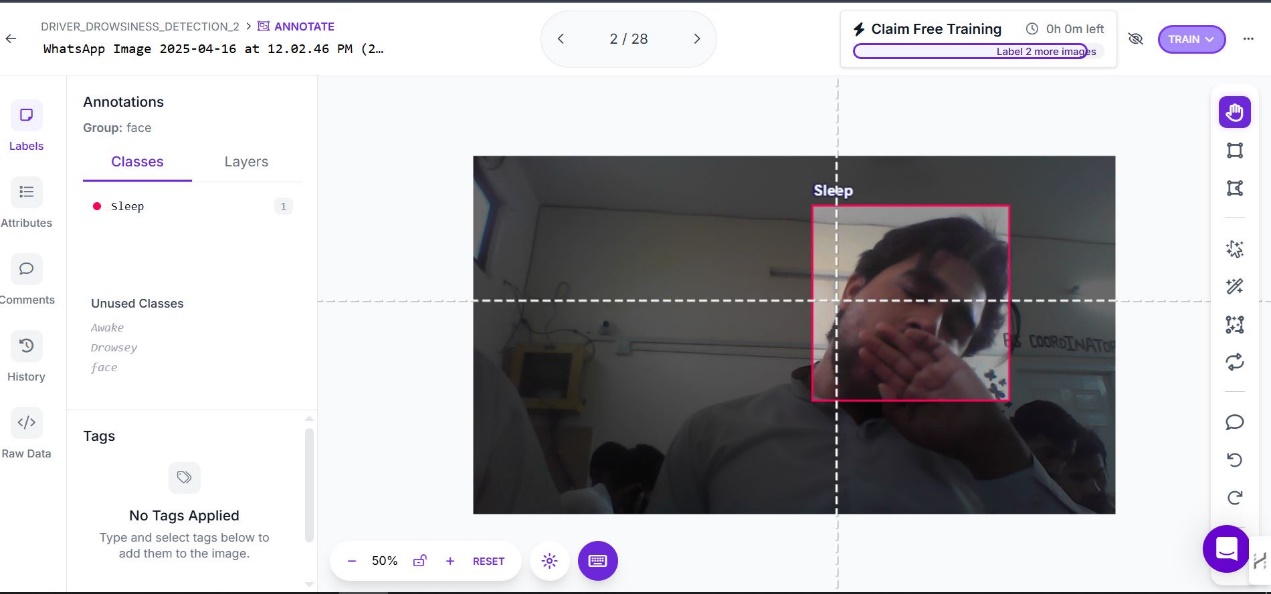


Figure 16: Annotating Images

* Apply **preprocessing steps** such as image resizing, auto-orientation correction, and format conversion (to YOLOv8-compatible .txt format).

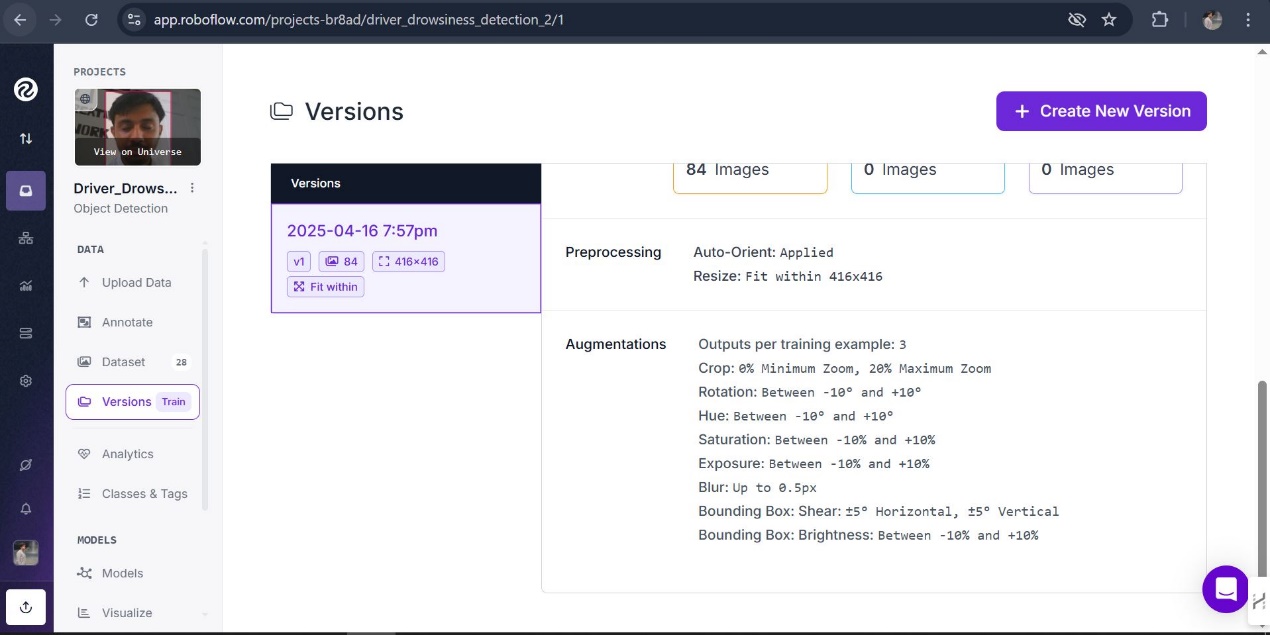


Figure 17: Preprocessing and Augmentation

* Export **the dataset** directly in YOLO format, making it easy to integrate with our training pipeline in Google Colab.

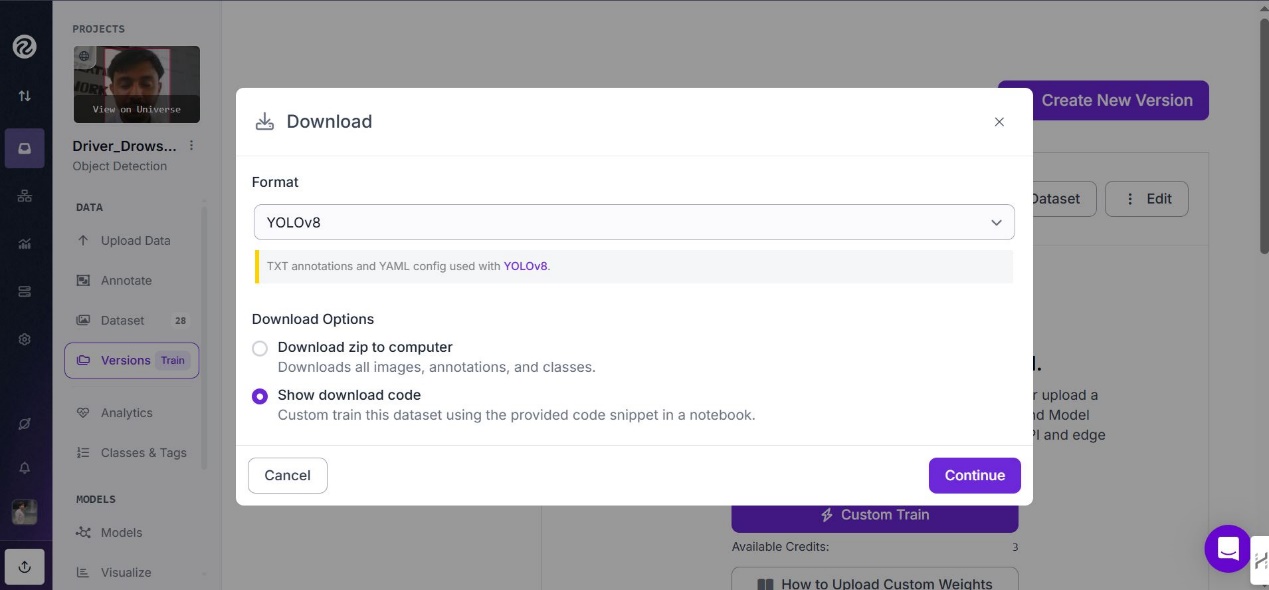


Figure 18: Download the dataset

### **4.2.4 Model Training**

* To train the YOLOv8-based drowsiness detection model efficiently, we utilized **Google Colaboratory (Google Colab)** as our cloud-based development environment. Google Colab provides free and paid access to high-performance GPUs and TPUs, enabling faster training and processing compared to standard local hardware.

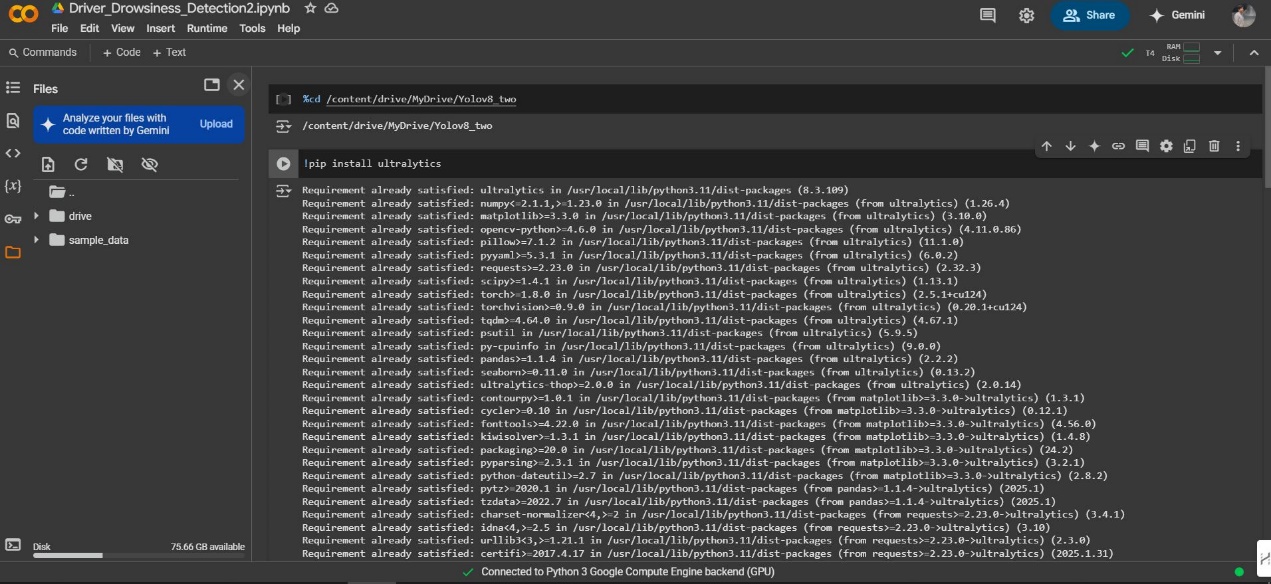


Figure 19: Importing libraries

* In order to train the YOLOv8 model on our custom dataset, we created a configuration file named **data.yaml**. This file plays a critical role in informing the training script about the structure and location of the dataset.

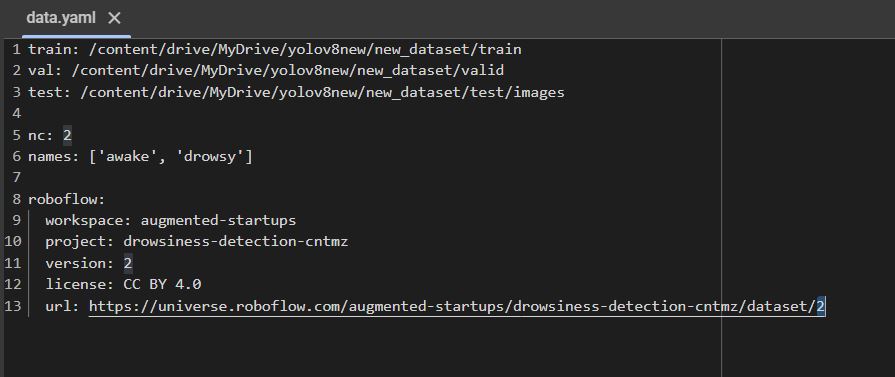


Figure 20: Data.yaml file

* For optimal learning and convergence of the YOLOv8 model, we trained the system for **150 epochs**. In deep learning, an epoch refers to one complete pass through the entire training dataset. Training over multiple epochs helps the model learn patterns more accurately by adjusting weights incrementally with each iteration.

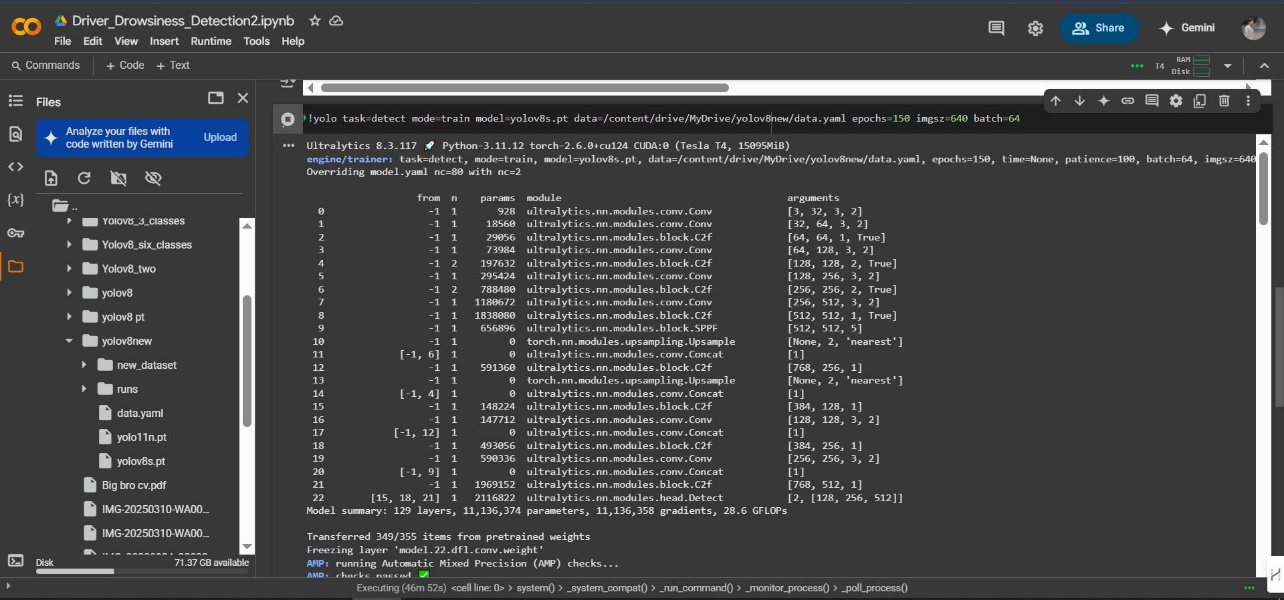


Figure 21: Training on epochs

# 4.3 GUI Development:

To make the driver drowsiness detection system easy to use and accessible for everyone, a Graphical User Interface (GUI) was developed. The GUI allows the user to interact with the system through simple controls such as starting or stopping the detection process without requiring any technical knowledge.

* We use Vs code to develop the interface and load our trained model path in it.
* We also text-speech back feature is added, that tells either the person state (Awake, drowsey).

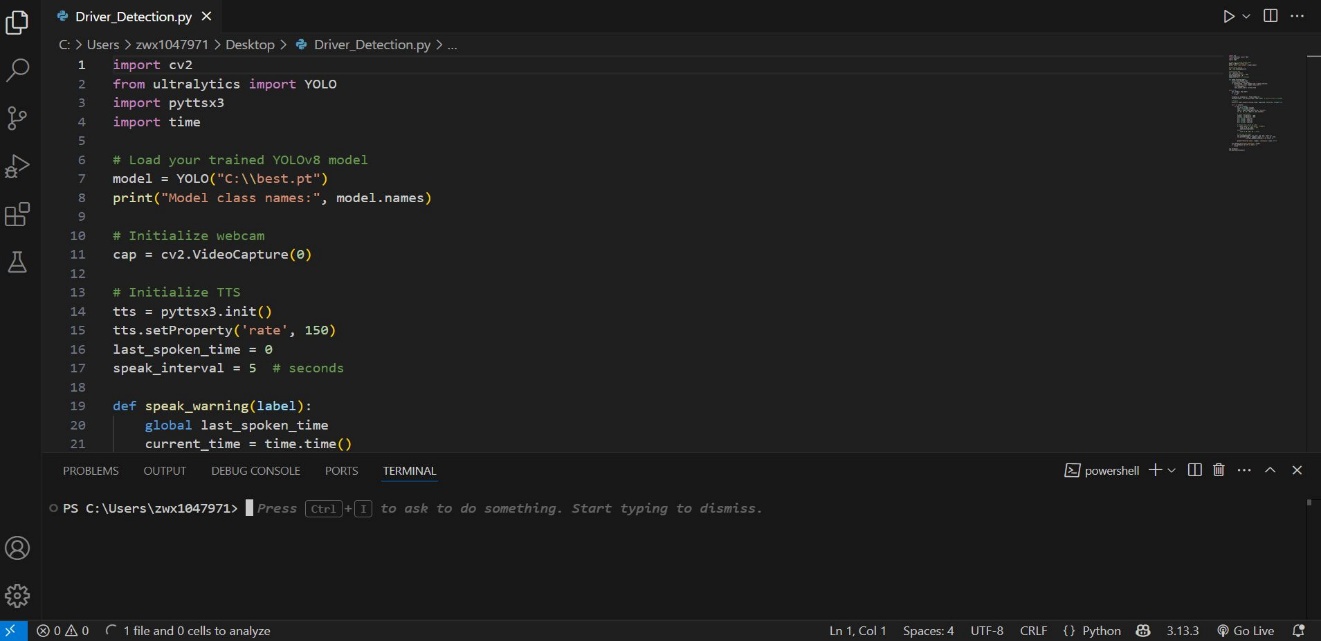


Figure 22: Loading the model

* Tkinter was chosen for its simplicity and seamless integration with libraries like OpenCV and pyttsx3, ensuring smooth project operation.
* The minimalistic interface includes Start and Stop buttons, with real-time status updates for a clean, user-friendly experience.

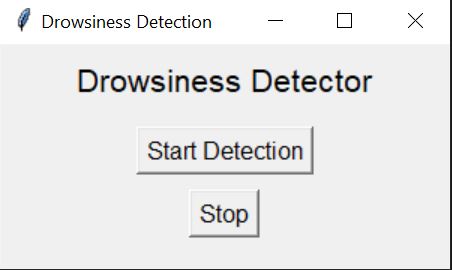


Figure 23: GUI interface

# 4.5 Testing and Results

* For testing we check our project in live-detection.

**

Figure 24: live detection

* The detection results clearly demonstrate the model’s ability to accurately classify driver states as either "awake" or "drowsy" with high confidence in real-time scenarios.



Figure 25: Classification output

* The confusion matrix demonstrates the model’s high accuracy, correctly classifying "awake," "drowsy," and "background" states with minimal misclassification

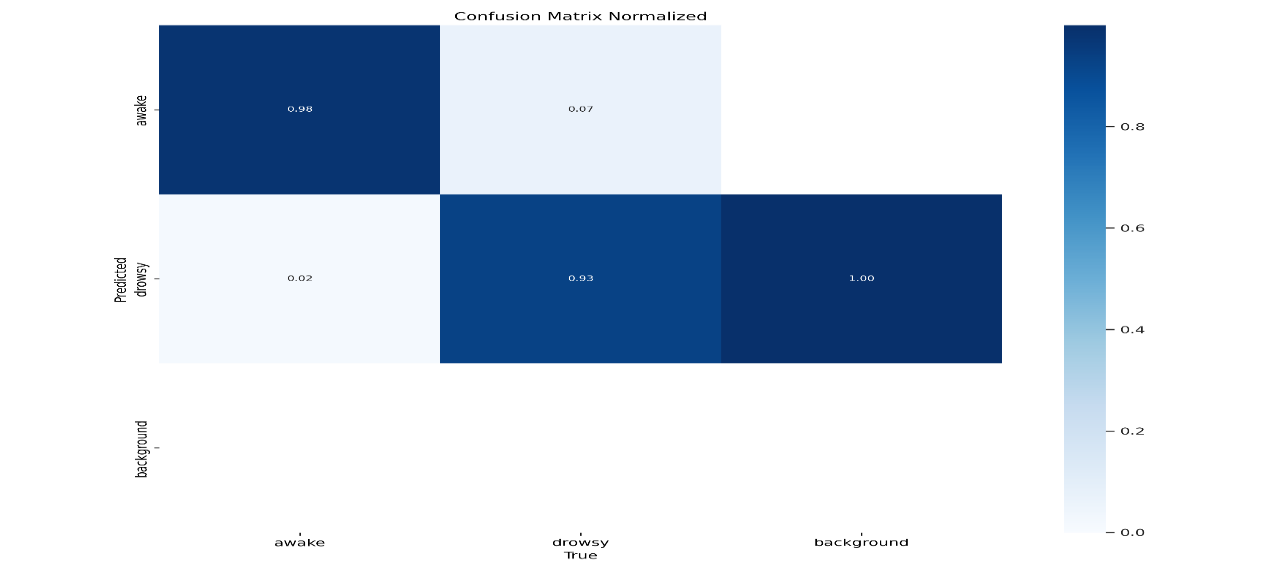


Figure 26: Confusion matrix

* The model shows effective learning with decreasing losses (box, cls, dfl) and increasing performance metrics (precision, recall, mAP) over 150 epochs, stabilizing after ~50 epochs.

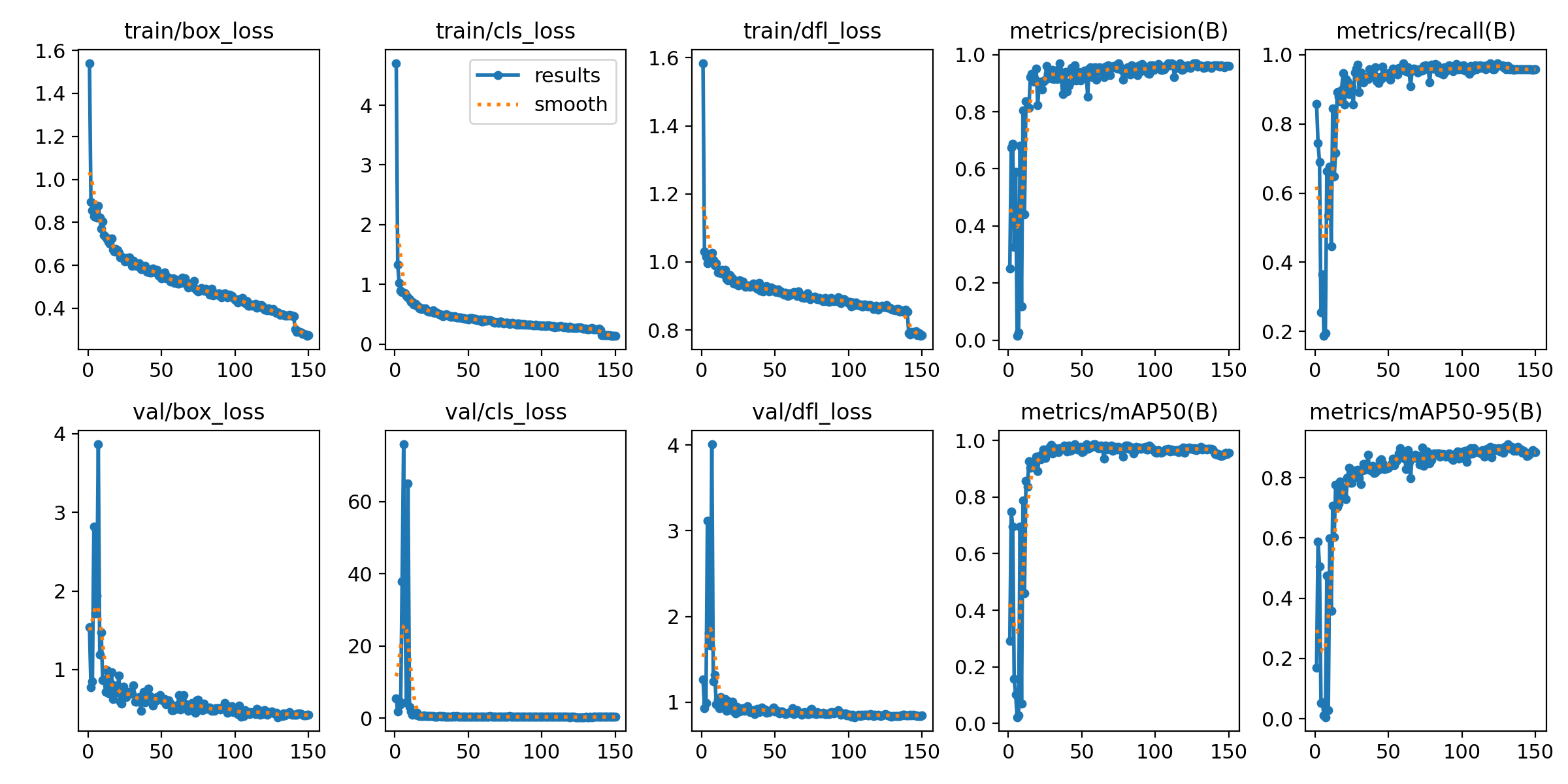


Figure 27: Performance Evaluation

* The Recall-Confidence Curve shows that the model maintains high recall (~1.0) for "awake" and "drowsy" classes up to a confidence threshold of ~0.9, with an overall recall of 0.97 at a confidence of 0.000

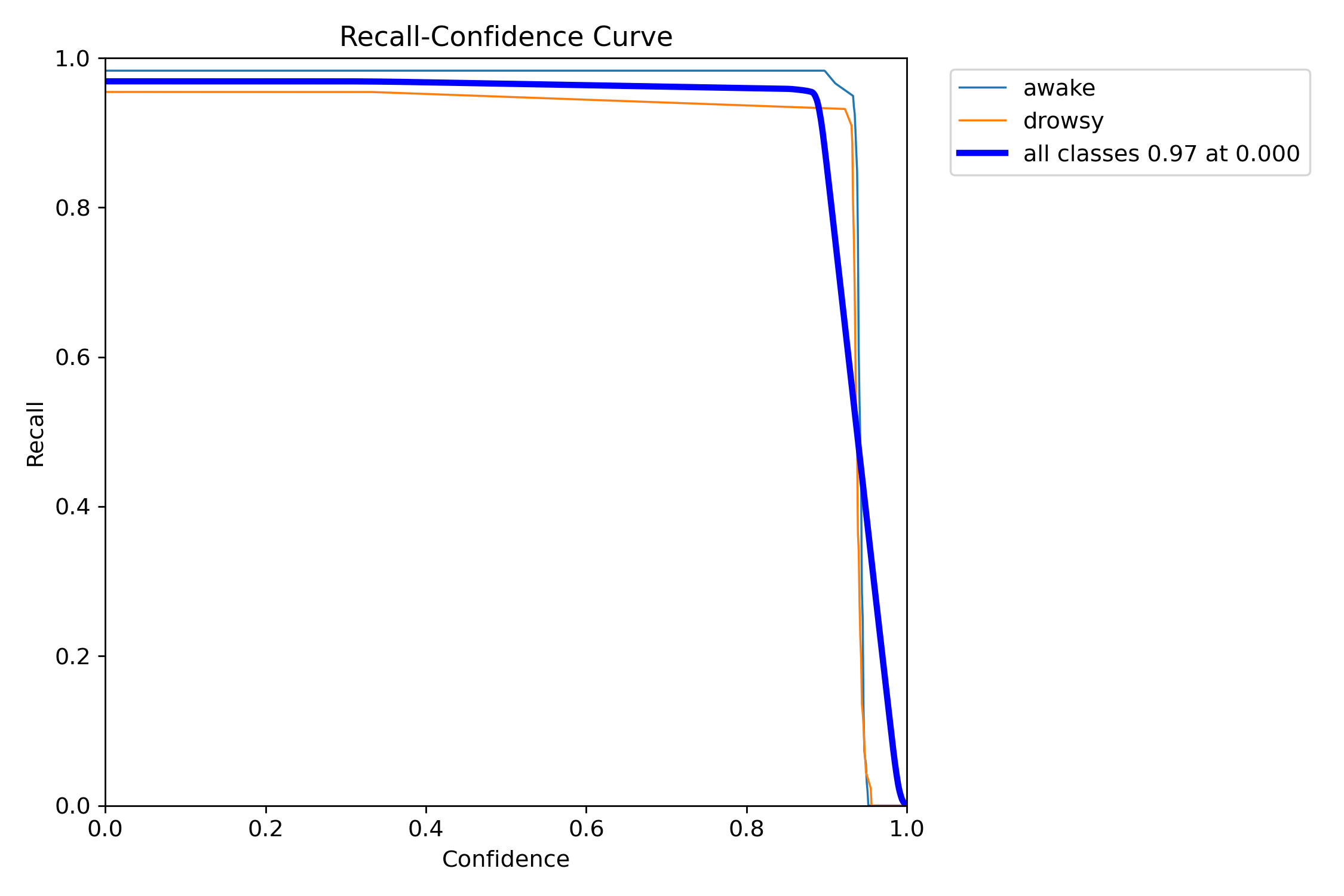


Figure 28: Recall vs Confidence curve

* The F1-Confidence Curve shows that the model maintains a high F1 score (~1.0) for "awake" and "drowsy" classes up to a confidence threshold of ~0.9, with an overall F1 of 0.96 at a confidence of 0.365.

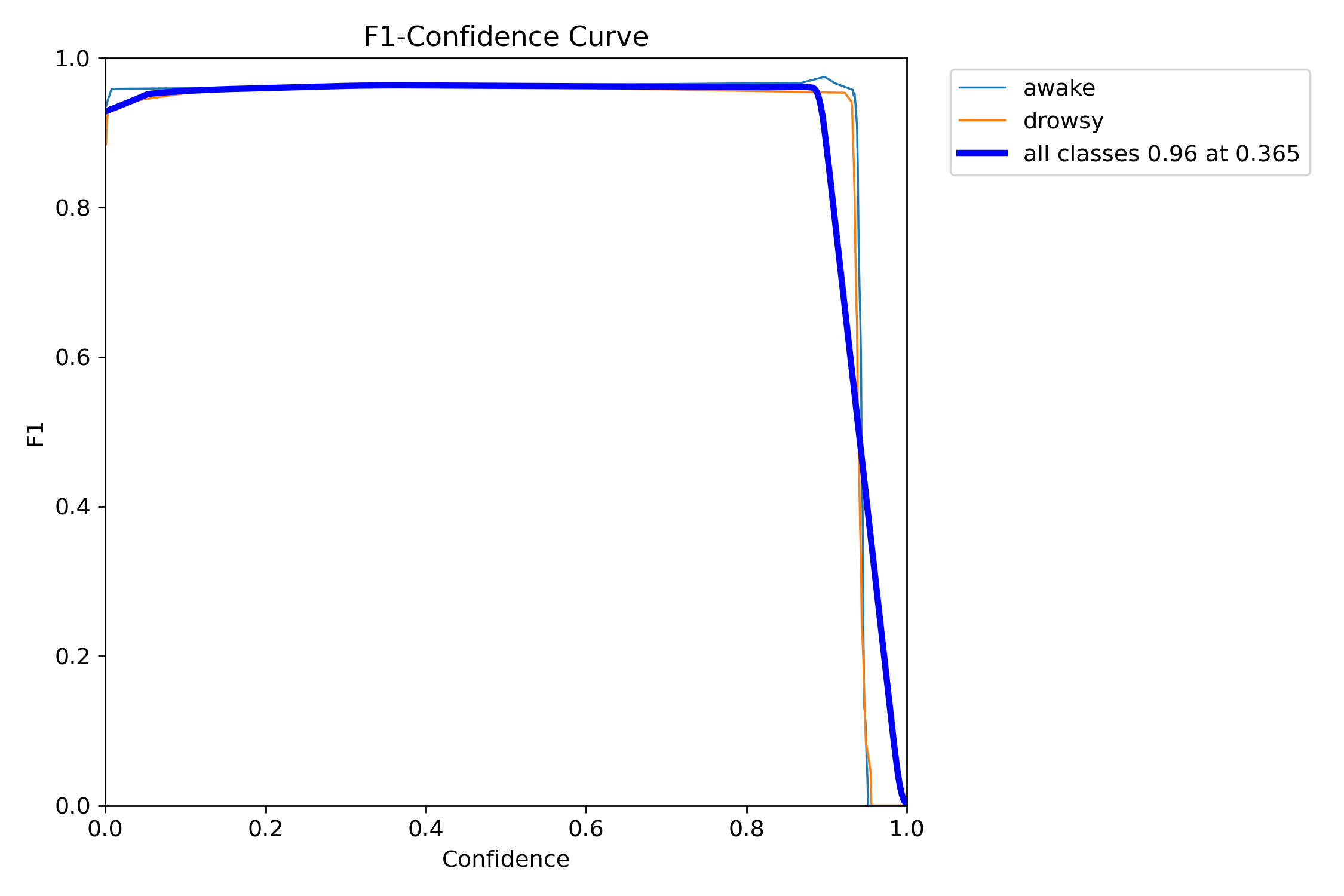


Figure 29: F1-curve

* The dataset has more "awake" instances (~600) than "drowsy" (~500), with "awake" bounding boxes generally larger (width 0.3-0.5, height 0.45-0.55) than "drowsy" (width 0.15-0.30, height 0.25-0.35).

Chapter 5

Conclusion and Future Work

## **5.1. Conclusion**

The driver drowsiness detection system is an emerging and highly impactful technology nowadays and is being used for a variety of purposes like driver monitoring, fatigue detection, and accident prevention. Driver monitoring technologies play a vital role in different fields including transportation safety, healthcare applications, security systems, and fleet management solutions.  
The drowsiness detection system aims to eliminate the major risks associated with driver fatigue, such as delayed reaction times, lack of focus, and ultimately, road accidents.  
We propose an AI-based driver drowsiness detection system that can be used as a reliable and affordable solution for improving road safety. In this system, drowsiness is detected through real-time face monitoring using a central database and deep learning models. We used Python programming language along with the OpenCV computer vision library for the development of the system. The system processes the driver’s facial landmarks, such as eye closure and yawning, to detect signs of drowsiness. There is also an alert mechanism to immediately warn the driver when signs of fatigue are detected.  
The results are automatically processed and displayed through the system’s interface, ensuring smooth and efficient operation. Nowadays, everyone wants solutions that can work fast and automatically without much human intervention. The driver drowsiness detection system fulfills these needs by reducing the risks, time, and costs associated with manual driver monitoring and contributes greatly towards safer roads and driving experiences.

## **5.2. Future Work**

In the future, we aim to enhance the driver drowsiness detection system by integrating more advanced technologies and deep learning models to improve the overall performance and accuracy.  
Future developments will not only focus on detecting drowsiness but also on monitoring various other driver states such as detecting if the driver is using a mobile phone while driving, talking with passengers, eating, or showing signs of distraction.  
By expanding the system’s capabilities to monitor multiple behaviors, we can create a more comprehensive driver monitoring solution that can further reduce the chances of accidents caused by distraction and negligence.  
In addition, the use of more powerful hardware, improved algorithms, and real-time alert mechanisms can make the system even faster and more reliable. Future versions of the system can also integrate with vehicle control systems to take preventive actions automatically if necessary, such as slowing down the vehicle in case of extreme drowsiness or distraction.

References

1. World Health Organization, 2019. *Global status report on road safety 2018*. World Health Organization.
2. Connor, J., Norton, R., Ameratunga, S., Robinson, E., Civil, I., Dunn, R., Bailey, J. and Jackson, R., 2002. Driver sleepiness and risk of serious injury to car occupants: population based case control study. *Bmj*, *324*(7346), p.1125.
3. Ji, Q., Zhu, Z. and Lan, P., 2004. Real-time nonintrusive monitoring and prediction of driver fatigue. *IEEE transactions on vehicular technology*, *53*(4), pp.1052-1068.
4. Ji, Q., Zhu, Z. and Lan, P., 2004. Real-time nonintrusive monitoring and prediction of driver fatigue. *IEEE transactions on vehicular technology*, *53*(4), pp.1052-1068.
5. G. Jocher, A. Chaurasia, and J. Qiu, “Ultralytics YOLOv8,” GitHub, 2023. [Online]. Available: <https://github.com/ultralytics/ultralytics>
6. G. Bradski, “The OpenCV Library,” Dr. Dobb's Journal of Software Tools, 2000. [Online]. Available: <https://opencv.org/>
7. E. Bisong, Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners. Berkeley, CA: Apress, 2019, pp. 59–64.
8. K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, “Joint face detection and alignment using multitask cascaded convolutional networks,” IEEE Signal Processing Letters, vol. 23, no. 10, pp. 1499–1503, 2016.