DROWSINESS DETECTION SYSTEM

USING MACHINE LEARNING

A PROJECT REPORT

Submitted by

GOKUL R **210419106034**
JAYAPRAKASH M **210419106047**

in partial fulfillment for the award of the degree

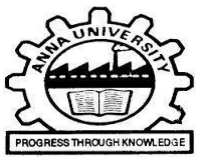
of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

CHENNAI INSTITUTE OF TECHNOLOGY (AUTONOMOUS)
(Affiliated to Anna University, Chennai)
CHENNAI-600 069



ANNA UNIVERSITY: CHENNAI-600 025
MARCH-2021

ANNA UNIVERSITY: CHENNAI-600 025

BONAFIDE CERTIFICATE

Certified that this project report "**DROWSINESS DETECTION USING MACHINE LEARNING**" is the bonafide work of **GOKUL R (210419106034), JAYAPRAKASH M (210419106047)** who carried out the project work under my supervision.

SIGNATURE

Dr. R. MENAKA, Ph.D.
HEAD OF THE DEPARTMENT
Associate Professor
Department of Electronics and
Communication Engineering,
Chennai Institute of Technology,
Kundrathur,
Chennai-600069.

SIGNATURE

Mrs. P. Vijaya Sri, M.E. (Ph.D.)
SUPERVISOR
Assistant Professor
Department of Electronics and
Communication Engineering,
Chennai Institute of Technology,
Kundrathur,
Chennai-600069.

Certified that the above students have attend of viva voce during the exam held on…….....

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We express our gratitude to our Chairman **Shri. P. SRIRAM** and all trust members of Chennai institute of technology for providing the facility and opportunity to do this project as a part of our undergraduate course.

We are grateful to our Principal Dr**. A. RAMESH M.E**, **Ph.D.** for providing us the facility and encouragement during the course of our work.

We sincerely thank our Head of the Department, **Dr. R. MENAKA M.E, Ph.D.** Department of Electronics and Communication Engineering for having provided us valuable guidance, resources and timely suggestions throughout our work.

We sincerely thank our Project Guide, **Mrs. P. Vijaya Sri, M.E (Ph.D.)** Professor, Department of Electronics and Communication Engineering for having provided us valuable guidance, resources and timely suggestions throughout our work.

We would like to extend our thanks to our **Faculty coordinators of the Department of Electronics and Communication Engineering**, for their valuable suggestions throughout this project.

We wish to extend our sincere thanks to all **Faculty members of the Department of Electronics and Communication Engineering** for their valuable suggestions and their kind cooperation for the successful completion of our project.

We wish to acknowledge the help received from the **Lab Instructors of the Department of Electronics and Communication Engineering** and others for providing valuable suggestions and for the successful completion of the project.

ABSTRACT

The Drowsy driving is responsible for one out of every four car accidents. The most terrifying aspect is that drowsy driving isn't merely falling asleep behind the wheel. When a driver is not paying full attention to the road, drowsy driving can be as simple as a momentary episode of unconsciousness. Each year, drowsy driving causes about 71,000 injuries, 1,500 deaths, and $12.5 billion in financial losses. We feel it is critical to find a solution for sleepiness detection, especially in the early phases of the problem, to prevent accidents. We also feel that tiredness can have a negative impact on people in the workplace and in the classroom. Although sleep deprivation and college go hand in hand, drowsiness in the workplace, particularly when operating with heavy machinery, can lead to significant injuries akin to those caused by driving while inebriated .

Most of the traditional methods to detect drowsiness are based on behavioural aspects while some are intrusive and may distract drivers, while some require expensive sensors. Therefore, in this project, a light-weight, real time driver's drowsiness detection system is developed and implemented on Web application. The system is capable of detecting facial landmarks, computes Eye Aspect Ratio (EAR) and Eye Closure Ratio (ECR) to detect driver's drowsiness based on adaptive thresholding. Machine learning algorithms have been employed to test the efficacy of the proposed approach. Empirical results demonstrate that the proposed model is able to achieve accuracy of 84% by YOLO v5 classifier.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ML	Machine Learning
API	Application Programming Interface
UI	User Interface
AI	Artificial Intelligence
CSV	Comma Separated Values
ROC	Receiver Operating Characteristics

CHAPTER 1

INTRODUCTION

Driving when tired has become one of the leading causes of traffic accidents. Drivers who drive at night or for lengthy periods of time without stopping are more likely to be in an accident. Because of this, a large number of fatal injuries and deaths occur. As a result, it has become a popular research topic. Physiological features, behavioral patterns, and vehicle-based features are all used in various systems for this purpose. Electroencephalogram (EEG), Electrooculogram (EOG), Electrocardiogram (ECG), heartbeat, pulse rate, and other physiological parameters are taken into account. The visual behaviors of drive that are studied here are eye blinking, eye closure, yawning, head bending, and so on. Wheel movement, acceleration, vehicle speed, braking pattern, divergence from lane pattern, and other vehicle-based variables are examples. The majority of these approaches are time-consuming and costly. We offer an alternative technique that uses photos to identify tiredness utilizing machine learning in this system.

1.1 OBJECTIVE

Primary Objective :

The primary objective of a drowsiness detection system is to monitor and detect signs of drowsiness or fatigue in an individual in real-time. The system can be used in various applications such as driving, aviation, and industrial settings where drowsiness can pose a safety risk. By detecting drowsiness early, the system can alert the individual or operator to take corrective actions or take a break to avoid accidents and improve productivity. The system may use various methods such as monitoring eye movements, head position, heart rate, and brain activity to detect drowsiness. The primary objective of a drowsiness detection system is to improve safety and prevent accidents caused by drowsiness or fatigue.

Secondary Objective:

The secondary objective of a drowsiness detection system is to provide insights and data on the patterns and causes of drowsiness. By analyzing the data collected from the system, researchers and organizations can gain a better understanding of the factors that contribute to drowsiness and fatigue, such as sleep deprivation, medications, and lifestyle habits. This information can be used to develop interventions and strategies to prevent drowsiness and improve sleep quality. Additionally, the system can be used to monitor the effectiveness of interventions and evaluate their impact on reducing drowsiness and fatigue. The secondary objective of a drowsiness detection system is to provide insights and data to improve overall health and well-being, in addition to improving safety.

1.2 ABOUT DROWSINESS:

Drowsiness refers to feeling more sleepy than normal during the day. People who are drowsy may fall asleep in when they do not want to or at times which can lead to safety concerns. The most common causes of excessive daytime sleepiness are sleep deprivation, obstructive sleep apnea, and sedating medications. Other potential causes of excessive daytime sleepiness include certain medical and psychiatric conditions and sleep disorders, such as narcolepsy. Feeling sleepy throughout the day can interfere with your quality of life, possibly hurting your performance at work or keeping you from participating in daytime activities. Drowsiness can also increase your risk of falling, which can lead to injury and disability, and it can affect your ability to drive safely.

1.3 SLEEP DEPRIVATION AND FATIGUE:

Sleep deprivation refers to the condition where an individual does not get adequate or restful sleep, which can lead to a range of negative effects on physical, cognitive, and emotional health. Chronic sleep deprivation can increase the risk of developing various health problems, such as obesity, diabetes, cardiovascular disease, depression, and anxiety.

Fatigue, on the other hand, refers to a feeling of extreme tiredness, lack of energy, and exhaustion that may result from prolonged sleep deprivation or other factors, such as physical or mental exertion, illness, or medication.

Sleep deprivation and fatigue are closely related, as sleep deprivation can lead to fatigue, and fatigue can also disrupt sleep quality and quantity, creating a vicious cycle of sleep deprivation and fatigue. To prevent and manage sleep deprivation and fatigue, it is essential to prioritize good sleep hygiene, establish healthy sleep habits, and seek medical attention if necessary.

CHAPTER 2

LITERATURE REVIEW

There are many previous researches regarding driver drowsiness detection system that can be used as a reference to develop a real-time system on detecting drowsiness for drivers. There is also several method which use different approaches to detect the drowsiness signs.

2.1. Drowsiness and Fatigue

Antoine Picot et al, stated that drowsiness is where a person is in the middle of awake and sleepy state. This situation leads the driver to not giving full attention to their driving. Therefore, the vehicle can no longer be controlled due to the driver being in a semi-conscious state. According to Gianluca Borghini et al, mental fatigue is a factor of drowsiness and it caused the person who experiences to not be able to perform because it decreases the efficiency of the brain to respond towards sudden events.

2.2. Electroencephalography (EEG) for Drowsiness Detection

FIG 2.1

Electroencephalography (EEG) is a method that measures the brain electrical activity. It can be used to measure the heartbeat, eye blink and even major physical movement

such as head movement. It can be used on human or animal as subjects to get the brain activity. It uses a special hardware that place sensors around the top of the head area to sense any electrical brain activity.

Authors mentioned that from the method that has been implemented by the previous researcher to detect drowsiness signs, the EEG method is best to be applied for drowsiness and fatigue detection. In the method, EEG have four types of frequency components that can be analyzed, i.e. alpha (α), beta (β), theta (θ) and delta (δ). When the power is increased in alpha (α) and delta (δ) frequency bands, it shows that the driver is facing fatigue and drowsiness. The disadvantages of this method are, it is very sensitive to noise around the sensors. For example, when the person is doing the EEG experiment, the surrounding area must be completely silent. The noise will interfere with the sensors that detect the brain activity. Another disadvantage of this method is that even if the result might be accurate, it is not suitable to use for real driving application. Imagine when a person is driving and he is wearing something on his head with full of wires and when the driver moves their head, the wire may strip off from their place. Even though it is not convenient to be used for real-time driving but for experiment purposes and data collection, it is one of the best methods so far.

2.3. Drowsiness detection using face detection system

Drowsiness can be detected by using face area detection. The methods to detect drowsiness within face area are vary due to drowsiness sign are more visible and clear to be detected at face area. From the face area, we can detect the eyes location. From eyes detection, author stated that there are four types of eyelid movement that can be used for drowsiness detection. They are complete open, complete close, and in the middle where the eyes are from open to close and vice versa.

a) Open eye b) Close eye c) Processed close eye

FIG 2.2

The algorithm processes the images captured in grey-scale method; where the color from the images is then transformed into black and white. Working with black and white images is easier because only two parameters have to be measured. The author then performs the edge detection to detect the edges of eyes so that the value of eyelid area can be calculated. The problem occurring with this method is that the size area of eye might vary from one person to another. Someone may have small eyes and looks like it is sleepy but some are not. Other than that, if the person is wearing glasses, there is obstacle to detect eye region. The images that being captured must be in certain range from the camera because when the distance is far from the camera, the images are blurred.

2.4. PERCLOS (Percentage of Eye Closure)

Drowsiness can be captured by detecting the eye blinks and percentage of eye closure (PERCLOS). For eye blink detection, propose a method which learned the pattern of duration of eyelid closed. According to, 'this proposed method measures the time for a person closed their eyes and if they are closed longer than the normal eye blink time, it is possible that the person is falling asleep'. In , the author mentioned that 'nearly 310.3ms are the average of normal person eye blink'. PERCLOS method proposes that drowsiness is measured by calculating the percentage of the eyelid 'droops'. Sets of eye open and eye closed have been stored in the software library to be used as a parameter to differentiate either the eyes is fully open or fully closed. For eyelid to droops, it happened in much slower time as the person is slowly falling asleep. Hence, the transition of the driver's drowsy can be recorded. Thus,

PERCLOS method put a proportional value where when the eyes is 80% closed, which it is nearly to fully close, it assumed that the driver is drowsy. This method is not convenient to be used in real-time driving as it needs fix threshold value of eye opening for the PERCLOS method to perform accurately. Both methods to detect drowsiness using eye blink pattern and PERCLOS have the same problem where the camera need to be placed at a specific angle in order to get a good image of video with no disturbance of eyebrow and shadow that cover the eyes.

2.5. Yawning Detection Method

Drowsiness of a person can be observed by looking at their face and behavior. The author proposed a method where drowsiness can be detected by mouth positioning and the images were process by using cascade of classifier that has been proposed by Viola-Jones for faces. The images were compared with the set of images data for mouth and yawning. Some people will close their mouth by their hand while yawning. It is an obstacle to get good images if a person is closing their mouth while yawning but yawning is definitely a sign of a person having drowsiness and fatigue.

Fig 2.3

After gone through the research papers and the existing methods, this project proposed that eyes and yawning detection method will be used. Eye blink duration gives the data that the longer the

person’s close their eyes, the drowsier it will be considered. It is because when a person is in drowsy state; its eyes will be closed longer than the normal eye blink. Other than that, yawning is one of the symptoms of drowsiness where it is a normal human response when yawning is the sign that they feel drowsy or fatigue.

CHAPTER 3

SOFTWARE REQUIREMENTS

3.1 Python

Python is a high-level, interpreted programming language that is widely used for various purposes such as web development, data analysis, artificial intelligence, machine learning, and scientific computing. It was first released in 1991 by Guido van Rossum and has since then gained immense popularity due to its simplicity, ease of use, and vast libraries. Python's syntax is easy to read and learn, making it an excellent choice for beginners to programming. Additionally, its large standard library and the availability of third-party packages and modules make it a versatile language suitable for a wide range of applications. Python is an interpreted language, meaning that there is no need to compile code before running it. This makes it a more accessible language for people who may not have experience with compiling code. Python also has a REPL (Read-Eval-Print Loop) environment, which allows for quick experimentation with code snippets.One of Python's unique features is its use of indentation to denote code blocks, which makes the code more readable and consistent. Python also supports multiple programming paradigms, including object-oriented, functional, and procedural programming. Overall, Python is a powerful and versatile programming language that is used by developers across various fields. Its simplicity, ease of use, and large community make it an excellent choice for beginners and experienced programmers alike.

3.2 LIBRARIES

A) TORCH

Torch is an open-source machine learning library for the Python programming language, primarily developed and maintained by Facebook AI Research (FAIR). It is based on Lua and was originally developed by Ronan Collobert and Soumith Chintala in 2002.Torch provides a wide range of tools and libraries for building and training deep neural networks. It supports various types of neural network architectures, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and feedforward neural networks (FFNNs). One of the key features of Torch is its flexibility and ease of use. It provides an intuitive and simple API that allows users to easily define and train deep learning models.Additionally, Torch supports GPU acceleration, which can significantly speed up training and inference times. Torch also includes a number of pre-trained models, making it easy for developers to quickly build and deploy machine learning applications. It also provides a powerful debugging and visualization tool, allowing users to visualize and analyze the behavior of their models during training and inference. Overall, Torch is a powerful and flexible machine learning library that is widely used by researchers and developers in the field of artificial intelligence. Its ease of use and flexibility make it an excellent choice for building and training deep neural networks.

B) MATPLOTLIB

Matplotlib is a popular data visualization library in Python that provides a wide range of plotting functionalities. It is built on top of NumPy and provides a convenient interface for creating static, animated, and interactive plots in various formats, including line plots, scatter plots, bar plots, histograms, and more. Matplotlib is highly customizable, allowing users to

adjust every aspect of the plot, including colors, labels, legends, axes, and annotations. It also provides support for 3D visualization and geographic mapping. Matplotlib can be used in various environments, including Jupyter notebooks, Python scripts, and web applications. It is compatible with other Python libraries, such as Pandas, Seaborn, and Scikit-learn, making it a valuable tool in data analysis and machine learning. Overall, Matplotlib is a powerful and flexible library that enables data scientists and researchers to communicate insights and discoveries through visualizations effectively.

C) NUMPY

NumPy is a widely-used library in Python that provides support for large, multi-dimensional arrays and matrices, as well as a collection of mathematical functions to operate on them. It is an essential tool in scientific computing, data analysis, and machine learning. NumPy's core data structure is the ndarray (n-dimensional array), which is used to represent arrays of any dimensionality. NumPy arrays are more efficient and provide more functionality than built-in Python lists. NumPy also provides an extensive collection of mathematical functions for array operations, including linear algebra, Fourier analysis, and statistics. NumPy has several advantages, including faster computations and better memory efficiency. It also provides an interface to integrate with other scientific computing libraries, such as SciPy, Pandas, and Matplotlib. Overall, NumPy is an essential tool for scientific computing, data analysis, and machine learning applications in Python. It enables developers to perform complex computations on large datasets efficiently, making it an invaluable tool in the data science community.

D) OPEN CV

OpenCV (cv2) is an open-source library of programming functions mainly aimed at real-time computer vision. It is written in C++ and has interfaces for C++, Python, and Java. OpenCV is used for a variety of applications, including image and video processing, object detection and recognition, face detection and recognition, and machine learning. In Python, OpenCV is available as a Python package called "cv2." It provides many pre-built image and video processing functions, such as reading and writing images and videos, resizing and cropping images, applying filters and transformations, and detecting and tracking objects in videos. OpenCV is also useful for machine learning applications, as it provides tools for creating and training machine learning models. It has support for popular machine learning libraries, such as TensorFlow and Keras, and can be used for tasks such as object detection and classification. Overall, OpenCV is a versatile and powerful library that is widely used in computer vision and machine learning applications. Its Python interface, cv2, makes it easy for developers to use OpenCV in their Python-based projects.

E) PANDAS

Pandas is a popular open-source library for data manipulation and analysis in Python. It provides fast and efficient data structures, such as DataFrame and Series, for handling and analyzing large datasets. Pandas provides a wide range of data manipulation functions, such as merging, grouping, filtering, and transforming data. It can handle various data formats, such as CSV, Excel, SQL databases, and JSON, and supports data cleaning, data normalization, and data aggregation. One of the key features of Pandas is its ability to handle missing data and perform data alignment, making it a valuable tool for data pre-processing. It also provides

powerful data visualization capabilities, allowing users to create a wide range of plots and charts. Pandas is used in a wide range of applications, such as finance, economics, social science, and engineering. It is also widely used in machine learning and data science workflows, as it integrates well with other Python libraries, such as NumPy, Matplotlib, and Scikit-learn. Overall, Pandas is a powerful and flexible library that provides an efficient and easy-to-use interface for data manipulation and analysis in Python.

F) SCIPY

SciPy is an open-source library for scientific computing and technical computing in Python. It provides a wide range of scientific algorithms and functions for various applications, including signal and image processing, optimization, interpolation, integration, and statistics. SciPy builds on the NumPy library, adding additional functionality and convenience functions to make scientific computing tasks more accessible to non-expert users. Some of the core modules of SciPy include: scipy.integrate: Provides functions for numerical integration, including the quad, dblquad, and tplquad functions. scipy.optimize: Provides functions for optimization, including curve fitting, minimization, and root finding. scipy.signal: Provides functions for signal processing, including filtering, Fourier analysis, and wavelet transforms. scipy.interpolate: Provides functions for interpolation and extrapolation of data. scipy.stats: Provides statistical functions for probability distributions, hypothesis testing, and descriptive statistics. SciPy is widely used in scientific computing and data analysis, and its functionality is integrated into many other Python libraries, including Pandas and Scikit-learn. Overall, SciPy is a powerful and versatile library that provides a wide range of scientific algorithms and functions for various applications, making it an essential tool for scientific computing in Python.

G) FLASK

Flask is a lightweight open-source web application framework written in Python. It is designed to be simple, flexible, and easy to use, making it an excellent choice for small to medium-sized web applications. Flask provides a set of tools and libraries for building web applications, including routing, template rendering, and request handling. It also supports integration with other Python libraries, such as SQLAlchemy for database integration, and WTForms for form handling. One of the key features of Flask is its extensibility, allowing developers to easily add new functionality to their applications through the use of extensions. Flask has a vibrant and active community that has created a wide range of extensions for various applications, such as Flask-Security for authentication and Flask-RESTful for building REST APIs. Flask is also known for its micro-framework approach, which means that it provides only the essentials required to build a web application, making it easy to understand and lightweight. This approach also allows developers to choose the specific tools and libraries they need for their application, rather than being locked into a specific set of tools. Overall, Flask is an excellent choice for developers who want to build small to medium-sized web applications quickly and easily. Its simplicity, flexibility, and extensibility make it a popular choice for many Python developers.

3.3 TOOLS

A) VS CODE

Visual Studio Code (VS Code) is a free and open-source source code editor developed by Microsoft. It is designed to be lightweight, fast, and customizable, making it a popular choice

for many developers. VS Code supports a wide range of programming languages, including Python, JavaScript, C++, and many others, and provides features such as syntax highlighting, auto-completion, and debugging. It also has a built-in terminal and supports source control integration with Git. One of the key features of VS Code is its extensive extension marketplace, which allows developers to add additional functionality to the editor through extensions. There are thousands of extensions available for VS Code, including language support, debugging tools, and productivity tools. VS Code also provides a customizable user interface, allowing users to personalize the editor to their preferences. Users can choose from a variety of themes, icons, and key bindings, and can even create their own custom settings. Overall, VS Code is a powerful and versatile code editor that provides a range of features and customization options for developers. Its popularity is due in part to its ease of use, fast performance, and extensive extension marketplace, making it a great choice for developers of all skill levels.

B) JUPYTER NOTEBOOK

Jupyter Notebook is an open-source web application that allows users to create and share documents containing live code, equations, visualizations, and narrative text. It is popular among data scientists, researchers, and educators for its ease of use and interactive nature. Jupyter Notebook supports a wide range of programming languages, including Python, R, and Julia, and allows users to write and run code cells interactively. It also supports the creation of markdown cells, which allow users to include narrative text and formatted equations in their documents. One of the key features of Jupyter Notebook is its ability to display visualizations directly in the notebook interface. This allows users to create interactive plots, charts, and graphs that can be explored and manipulated directly within the notebook. Jupyter Notebook also supports the installation of additional packages and extensions, which can be used to extend its functionality. There are many third-party extensions available, including ones for debugging,

code formatting, and version control integration. Overall, Jupyter Notebook is a powerful and versatile tool for data analysis, research, and education. Its ability to combine code, narrative text, and visualizations in a single document makes it an ideal platform for communicating complex ideas and analyses.

C) ANACONDA NAVIGATOR

Anaconda Navigator is a desktop graphical user interface (GUI) that comes with the Anaconda distribution of Python. It is designed to simplify the process of managing data science packages, environments, and projects, allowing users to focus on their analysis and research. Anaconda Navigator provides a convenient interface for managing Python packages and environments, allowing users to easily install, update, and remove packages and switch between different environments. It also provides access to a range of popular data science packages, including NumPy, Pandas, Matplotlib, and Scikit-learn. In addition to package management, Anaconda Navigator provides tools for managing data science projects, including Jupyter Notebook, Spyder, and RStudio. These tools allow users to create, edit, and share analysis scripts and notebooks, and provide powerful tools for data visualization and exploration. One of the key features of Anaconda Navigator is its simplicity and ease of use. Its user-friendly interface makes it easy for beginners to get started with data science, while also providing advanced features for experienced users. Overall, Anaconda Navigator is a powerful and versatile tool for data science that provides a convenient interface for managing packages, environments, and projects. Its ease of use and comprehensive package management make it a popular choice for many data scientists and researchers.

CHAPTER 4

PROPOSED METHOD

A proposed system for drowsiness detection using the YOLOv5 algorithm and Torch would involve several components, including data collection, labeling, training, and testing.

The first step would be to collect data, which could involve capturing images or video of drivers in a vehicle, along with metadata such as time of day, vehicle speed, and other relevant information. This data would be used to train the drowsiness detection model.

Next, the images would need to be labeled using a tool such as LabelImg, which would involve manually drawing bounding boxes around the eyes and other facial features of the driver to indicate whether they are open or closed. This labeled data would be used to train the YOLOv5 model.

Once the data has been labeled, the YOLOv5 model would be trained using Torch, a deep learning framework that can be used to train and deploy neural networks. The training process would involve feeding the labeled images into the model and adjusting its weights and parameters until it accurately detects drowsiness.

Finally, the model would be tested using a set of images or video captured in real-world conditions, and its performance evaluated based on metrics such as accuracy, precision, and recall. The system could be integrated into a real-time drowsiness detection system that alerts drivers when they are at risk of falling asleep at the wheel.

4.1 IMPORTING DEPENDENCY AND CLONING REPOSITORIES

A) INSTALL AND IMPORT DEPENDENCIES

To import PyTorch, OpenCV, NumPy, Matplotlib, OS, and Time in Python, you

can use the following code:

```
import torch
import cv2
import numpy as np
import matplotlib.pyplot as plt
import os
import time
```

PyTorch is a popular deep learning framework that provides tools for building and training neural networks. OpenCV is a computer vision library that provides a wide range of image and video processing functions. NumPy is a library for working with arrays and matrices in Python. Matplotlib is a data visualization library that can be used to create a wide range of charts and plots. OS is a library for working with operating system functions, such as file and directory operations. Time is a library for working with time-related functions, such as timing the execution of code.

B) CLONING THE REQUIRED REPOSITORIES

To clone the YOLOv5/Ultralytics repository from Github, you can use the following command in the terminal:

```
!git clone https://github.com/ultralytics/yolov5.git
```

This will create a local copy of the repository on your computer, which you can use to access

the YOLOv5 algorithm and associated code.

Similarly, to clone the LabelImg repository from Github, you can use the following command:

!git clone https://github.com/tzutalin/labelImg.git

This will create a local copy of the repository, which contains the code for a graphical image annotation tool that can be used to label images for object detection models such as YOLOv5.

4.2 CREATING CUSTOM IMAGES AND LABELLING

Creating new custom images of faces involves capturing images of faces from different angles and positions. These images can then be used to train machine learning models to recognize and classify different facial features.

To label these images using the LabelImg tool, you can use the cv2 and os libraries in Python.

The cv2 library can be used to capture images from a webcam or other camera, while the os library can be used to save these images to a directory on your computer.

LabelImg is an open-source graphical image annotation tool used to label images for object detection tasks in machine learning. It provides an easy-to-use interface that allows users to draw bounding boxes around objects of interest in an image and assign labels to these boxes. The tool supports multiple image formats including JPEG, PNG, and BMP, and can be used for both binary and multi-class object recognition tasks. LabelImg can generate output in various formats such as Pascal VOC and YOLO, which are commonly used in deep learning models. The software is written in Python and is available for Windows, Mac, and Linux platforms. Its ease of use, flexibility, and compatibility make it a popular choice for researchers and developers working on object detection projects.

The LabelImg tool is then opened to allow the user to label the image, and the labeled image is saved to the same directory. This process can be repeated to capture and label multiple images for training machine learning models.

In our case, we use yolo model to label the multiple images of the custom data.

4.3 TRAINING AND MODEL BUILDING

A) FINE TRAINING THE LABELLED DATASET

Fine-tuning a drowsiness model using YOLOv5 and PyTorch using custom images after labeling through LabelImg tool involves the following steps:

Data Collection: Collect a dataset of driver images with both open and closed eyes. The dataset should include a wide range of images, including different drivers and lighting conditions.

Labeling: Use LabelImg tool to label the images. This involves drawing bounding boxes around the eyes of the drivers in the images and labeling them as open or closed.

Preprocessing: Before training the deep neural network, the images should be preprocessed by resizing them to a fixed size, normalizing the pixel values, and converting them to a format that can be read by the YOLOv5 model.

Creating dataset.yaml file: Create a dataset.yaml file that specifies the location of the images, the class names, and the number of images for each class.

Training: Fine-tune the YOLOv5 model on the dataset using PyTorch. This involves loading

the pre-trained YOLOv5 model, replacing the final classification layer with a new layer for binary classification, and training the model using the new dataset.

Real-time detection: Once the model is trained, it can be used to detect drowsiness in real-time. This involves capturing video from a camera mounted in the car and feeding it into the model. The model will analyze each frame of the video and classify the state of the driver's eyes as open or closed. If the model detects that the driver's eyes are closed for an extended period of time, it can trigger an alert to warn the driver to wake up.

Now let's classify each parameter of the given code and understand its functionality:

!cd yolov5 && python train.py --img 416 --batch 16 --epochs 200 --data dataset.yml --weights yolov5s.pt --workers 2

!cd yolov5: This command changes the working directory to the yolov5 folder**.**

python train.py: This command runs the train.py file in the yolov5 folder.

--img 416: This parameter sets the input image size to 416 x 416. Smaller image sizes are faster but less accurate.

--batch 16: This parameter sets the batch size to 16. The batch size is the number of images processed in each iteration of the training process.

--epochs 200: This parameter sets the number of epochs to 200. An epoch is one iteration through the entire training dataset.

--data dataset.yml: This parameter specifies the location and configuration of the dataset. The dataset.yml file contains this information.

--weights yolov5s.pt: This parameter specifies the location and name of the pre-trained YOLOv5 model. The model will be fine-tuned using the custom dataset.

--workers 2: This parameter sets the number of worker threads used for data loading during training. Increasing the number of workers can speed up training, but it may also require more memory.

In conclusion, fine-tuning a drowsiness model using YOLOv5 and PyTorch using custom images after labeling through LabelImg tool involves collecting and labeling the dataset, preprocessing the images, creating a dataset.yml file, training the model, and using it for real-time detection. The provided code fine-tunes the YOLOv5 model on the custom dataset with specific parameters such as image size, batch size, number of epochs, dataset location and configuration, pre-trained model location and name, and number of worker threads.

B) LOADING THE MODEL TO PYTORCH

Here is a classification of the loading parameters in the given code:

```
model = torch.hub.load('ultralytics/yolov5', 'custom',
path='yolov5/runs/train/exp15/weights/last.pt', force_reload=True)
```

torch.hub.load: This function loads a pre-trained model from a repository using the provided

arguments.

'ultralytics/yolov5': This is the repository name that contains the pre-trained YOLOv5 models.

'custom': This is the name of the YOLOv5 variant that we want to load. 'custom' refers to a variant that has been trained on a custom dataset.

path='yolov5/runs/train/exp15/weights/last.pt': This parameter specifies the path to the trained model file that we want to load. In this case, the file is located in the yolov5/runs/train/exp15/weights/ directory and is named last.pt.

force_reload=True: This parameter forces the model to be reloaded from the specified path, even if it has already been loaded before. This can be useful if the model file has been updated since the last time it was loaded.

CHAPTER 5

RESULTS AND DISCUSSIONS

5.1 MODEL EVALUATION:

1. **Confusion matrix:** A confusion matrix is a table used to evaluate the performance of a classification model. It shows the number of true positives, true negatives, false positives, and false negatives predicted by the model. The true positive (TP) and true negative (TN) values represent the number of correct predictions made by the model, while the false positive (FP) and false negative (FN) values represent the number of incorrect predictions made by the model.

2. **F1 curve:** F1 curve is a plot of F1 scores against a threshold value used to binarize continuous scores into binary predictions. The F1 score is the harmonic mean of precision and recall, which measures the model's accuracy and completeness, respectively. The F1 score ranges from 0 to 1, where a value of 1 indicates perfect precision and recall.

3. **P_curve:** P_curve is a plot of precision values against the recall values used to evaluate the performance of a classification model. Precision measures the proportion of true positives predicted by the model out of all the positives predicted, while recall measures the proportion of true positives predicted out of all the actual positives in the dataset.

4. **PR_curve:** PR_curve is a plot of precision and recall values used to evaluate the performance of a classification model. Precision measures the proportion of true

positives predicted by the model out of all the positives predicted, while recall measures the proportion of true positives predicted out of all the actual positives in the dataset.

5. **R_curve:** R_curve is a plot of recall values against the threshold value used to binarize continuous scores into binary predictions. Recall measures the proportion of true positives predicted by the model out of all the actual positives in the dataset.

5.2 MODEL SELECTION

WHY YOLOv5 ?

YOLOv5 is a popular object detection framework that is known for its high accuracy, fast inference speed, and ease of use. These factors make it an excellent choice for a drowsiness detection system. Here are some of the reasons why YOLOv5 is considered the best for drowsiness detection:

High accuracy: YOLOv5 uses a highly optimized architecture that can detect objects with high accuracy, even in challenging environments. This is important for a drowsiness detection system, as it needs to be able to detect the subtle facial cues that indicate drowsiness accurately.

Fast inference speed: YOLOv5 is optimized for speed, and it can perform real-time object detection on a GPU, even on low-end hardware. This is important for a drowsiness detection system, as it needs to be able to detect drowsiness quickly to alert the driver or operator.

Easy to use: YOLOv5 has a user-friendly interface that makes it easy to train and deploy models, even for those with little to no experience in machine learning. This is important for a drowsiness detection system, as it needs to be easy to set up and use.

Pre-trained models: YOLOv5 has a large number of pre-trained models that can be fine-tuned on custom datasets quickly. This is important for a drowsiness detection system, as it allows developers to train a model on a small dataset of labeled images.

Flexibility: YOLOv5 is a flexible framework that can be customized to suit the specific needs of a drowsiness detection system. For example, it can be trained to detect drowsiness in different environments, such as in a car or in an industrial setting.

Overall, YOLOv5 is an excellent choice for a drowsiness detection system, as it offers high accuracy, fast inference speed, ease of use, pre-trained models, and flexibility.

YOLOv5

YOLOv5 is an object detection algorithm that stands for "You Only Look Once version 5". It is the latest version of the YOLO series and is an improvement over previous versions in terms of accuracy, speed, and ease of use. YOLOv5 is a deep learning-based approach for real-time object detection, tracking, and classification in images and videos.

YOLOv5 is based on a convolutional neural network (CNN) architecture that uses a single neural network to predict the bounding boxes and class probabilities of objects in an image. The algorithm divides the input image into a grid and applies object detection on each grid cell. Each cell predicts bounding boxes and class probabilities for the objects contained within the cell.

The YOLOv5 architecture consists of a backbone network, neck network, and head network. The backbone network is a pre-trained convolutional neural network (CNN) used for feature extraction. The neck network is an optional network that can be added to improve the feature representation. The head network is responsible for predicting the bounding boxes and class probabilities of objects.

YOLOv5 uses a range of data augmentation techniques, such as random scaling, flipping, rotation, and color jittering, to improve the model's robustness to variations in input data. It also uses a focal loss function that assigns more weight to hard-to-detect objects during training, improving the accuracy of the model.

YOLOv5 is known for its speed and accuracy, achieving state-of-the-art performance on popular object detection benchmarks such as COCO, PASCAL VOC, and OIDv4. It is implemented in the PyTorch deep learning framework and is open-source, making it accessible for researchers and developers to use and modify.

RESULTS REPORT OBTAINED FROM THE TRAINED MODEL

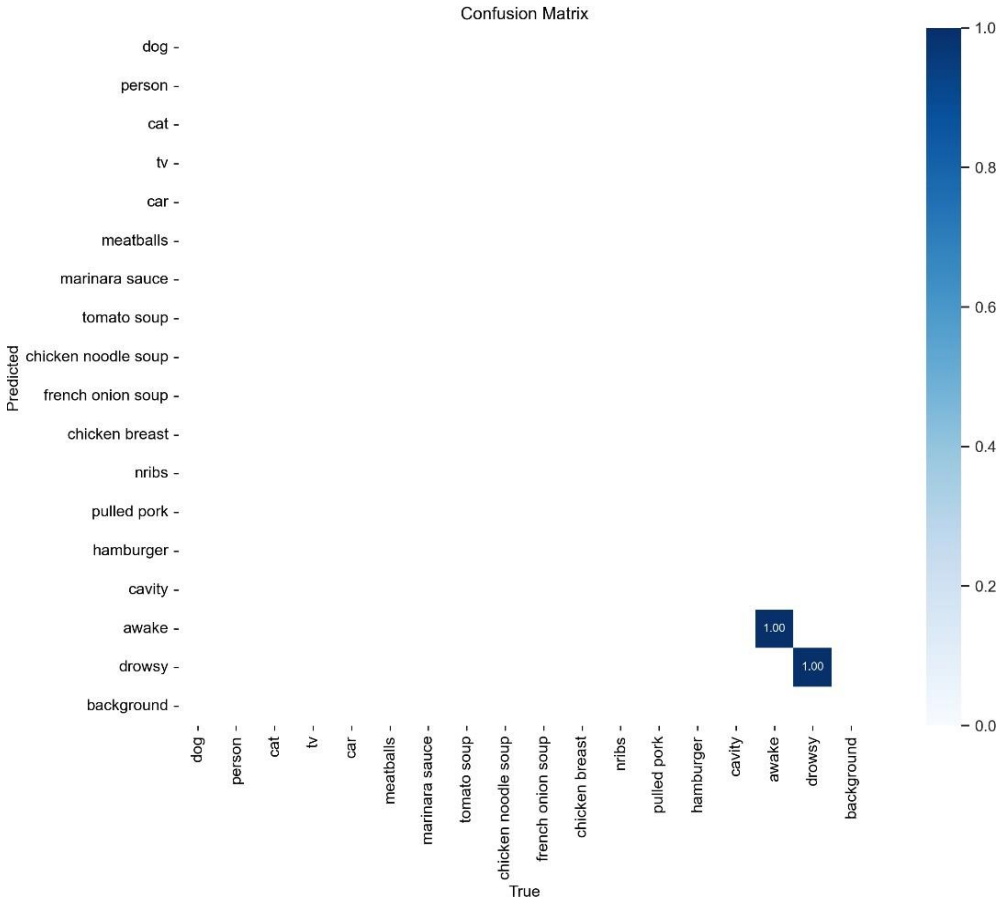
When training a YOLOv5 model for drowsiness detection, the number of epochs refers to the number of times the model has been trained on the dataset.

When a YOLOv5 model is trained for 200 epochs, it means that the model has gone through the dataset 200 times. During each epoch, the model updates its weights based on the loss computed during the forward and backward passes. This process of updating the weights helps the model learn and improve its accuracy.

The specific details of a YOLOv5 model trained for 200 epochs will depend on the dataset, the hardware used for training, and the specific configuration used for training. However, in general, a YOLOv5 model trained for 200 epochs will have undergone extensive training and should have a high level of accuracy for detecting drowsiness.

To evaluate the performance of a YOLOv5 model trained for 200 epochs, metrics such as precision, recall, and F1 score can be computed. These metrics can provide insight into how well the model is able to detect drowsiness and can be used to fine-tune the model further if necessary.

EXPERIMENT RESULTS AFTER TRAINING



+

FIG 5.1 CONFUSION MATRIX

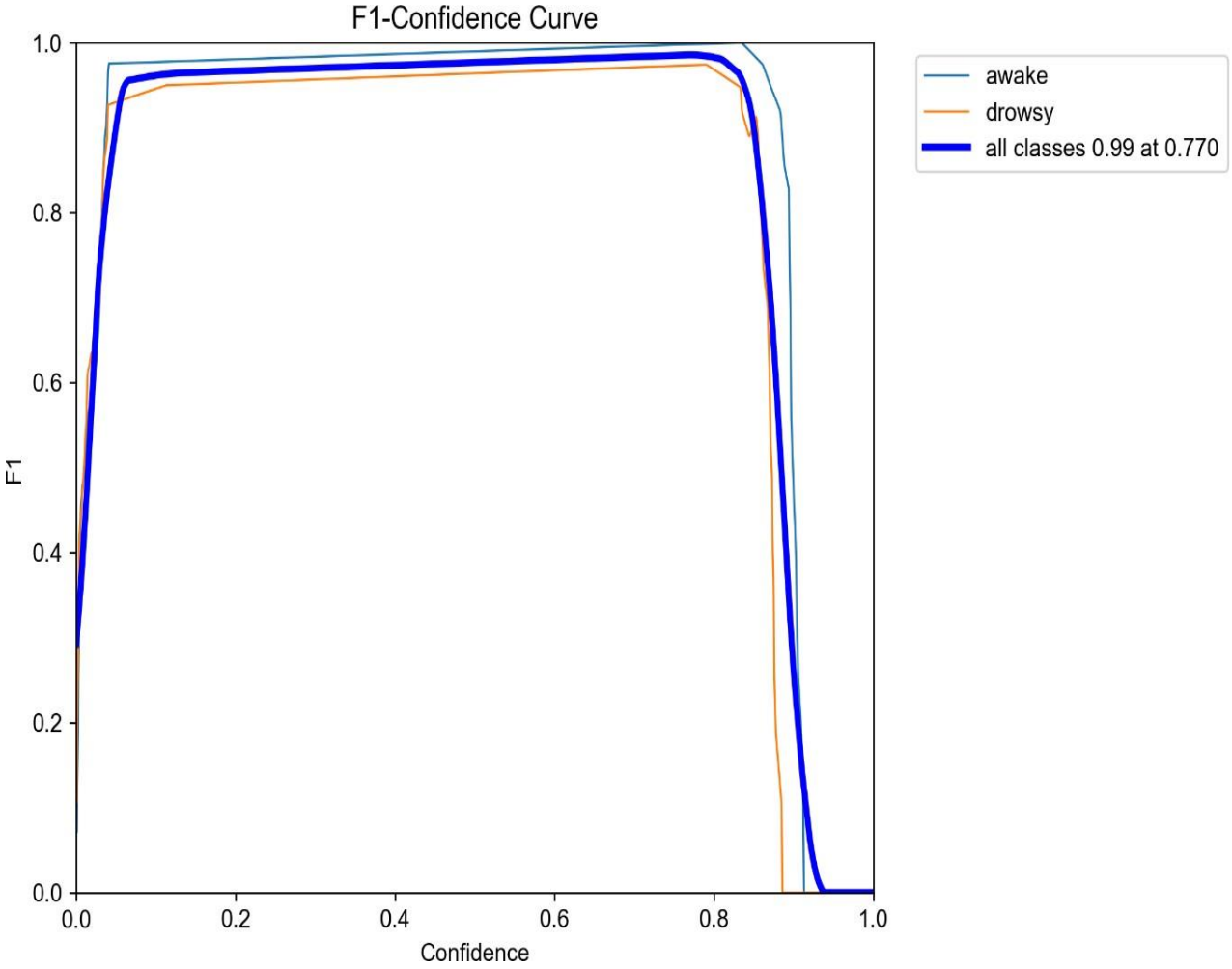


FIG 5.2 F1_CURVE

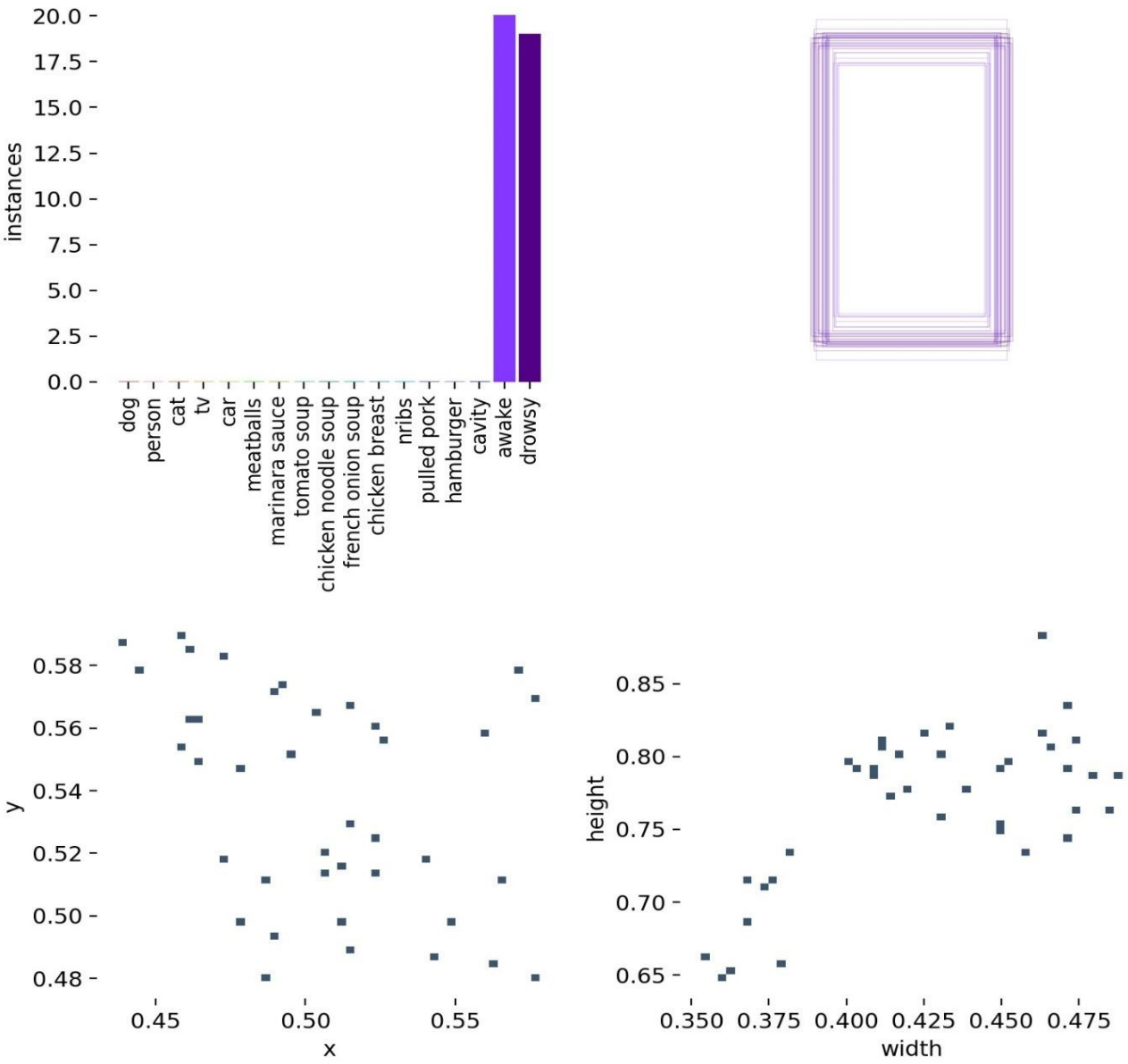


FIG 5.3 LABELS

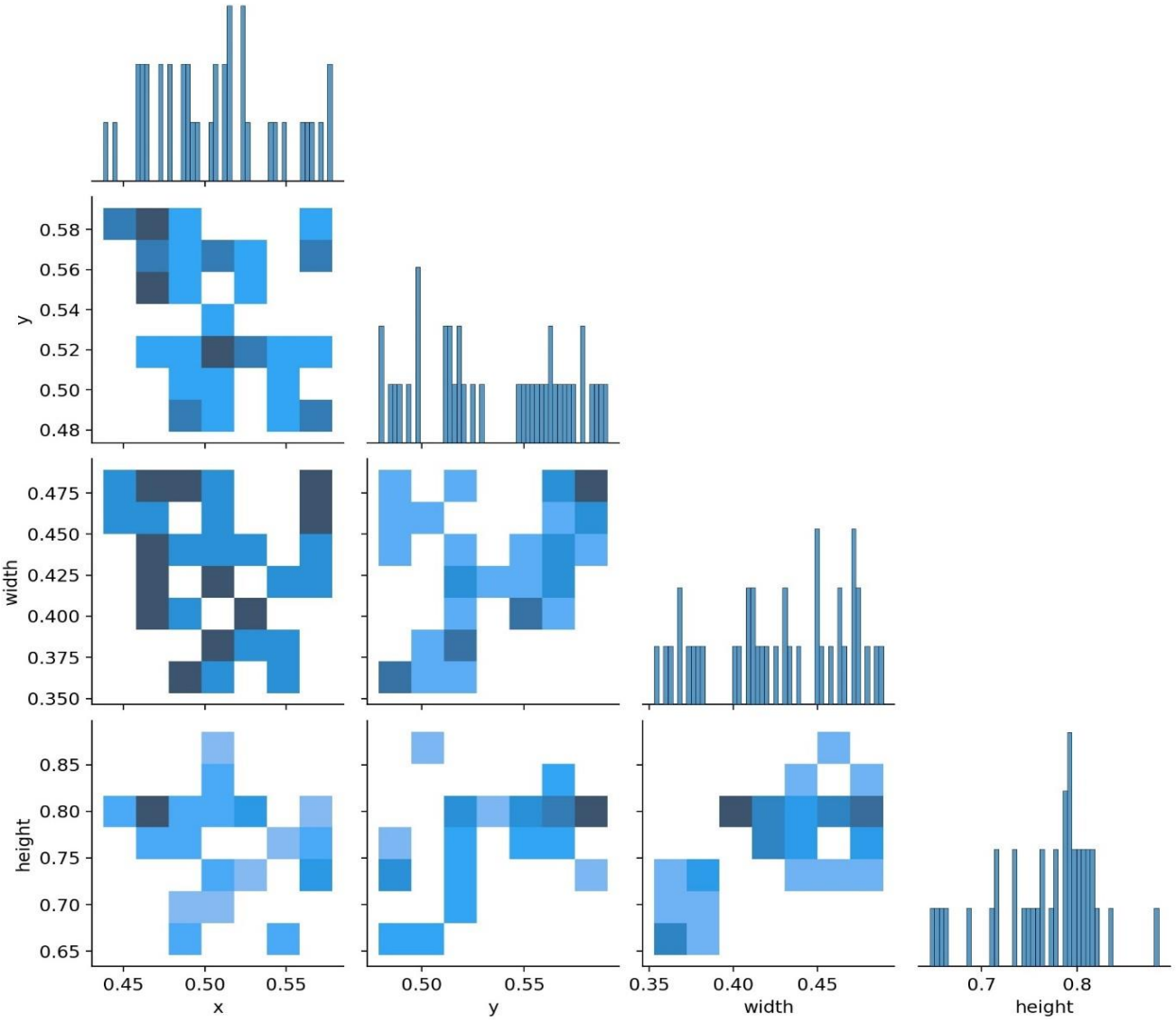


FIG 5.4 lABELS_CORRELOGRAM

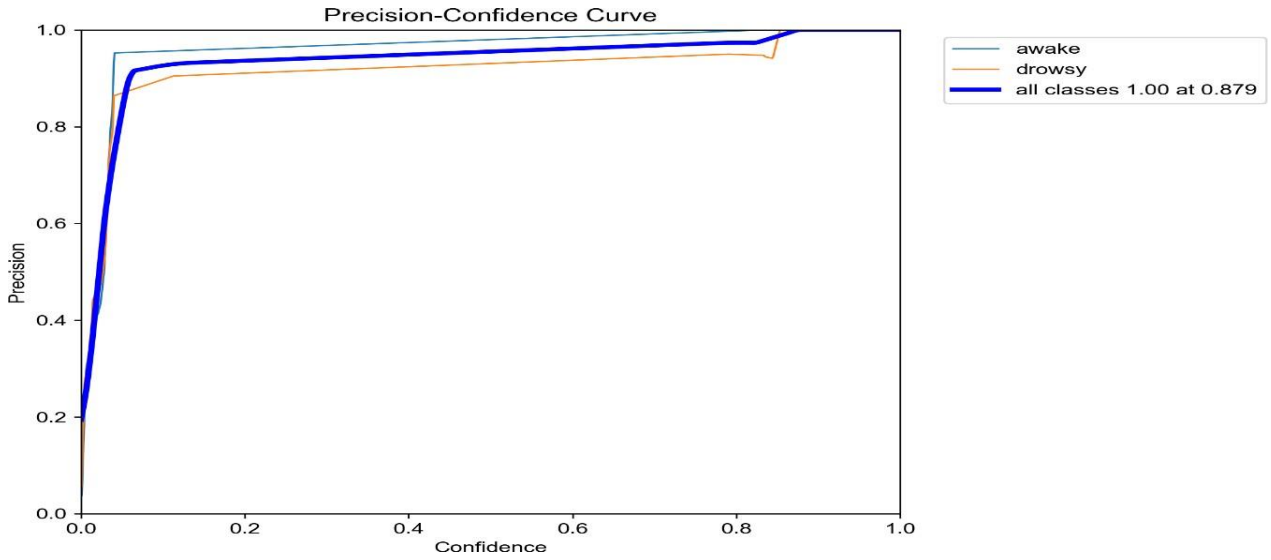


FIG 5.5 P_CURVE

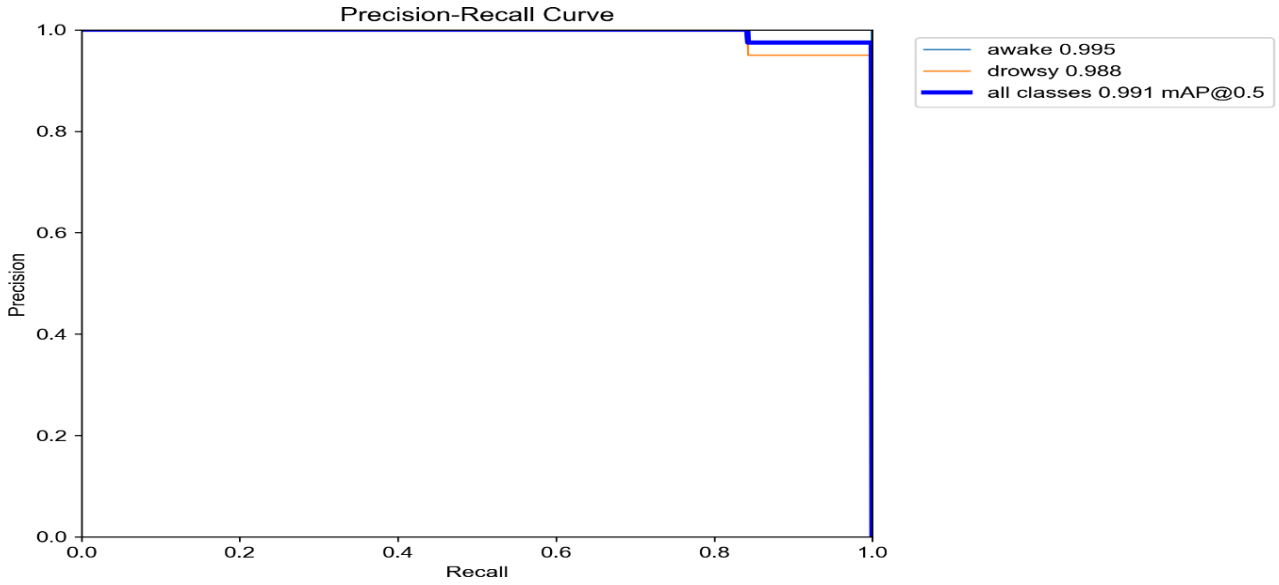


FIG 5.6 PR_CURVE

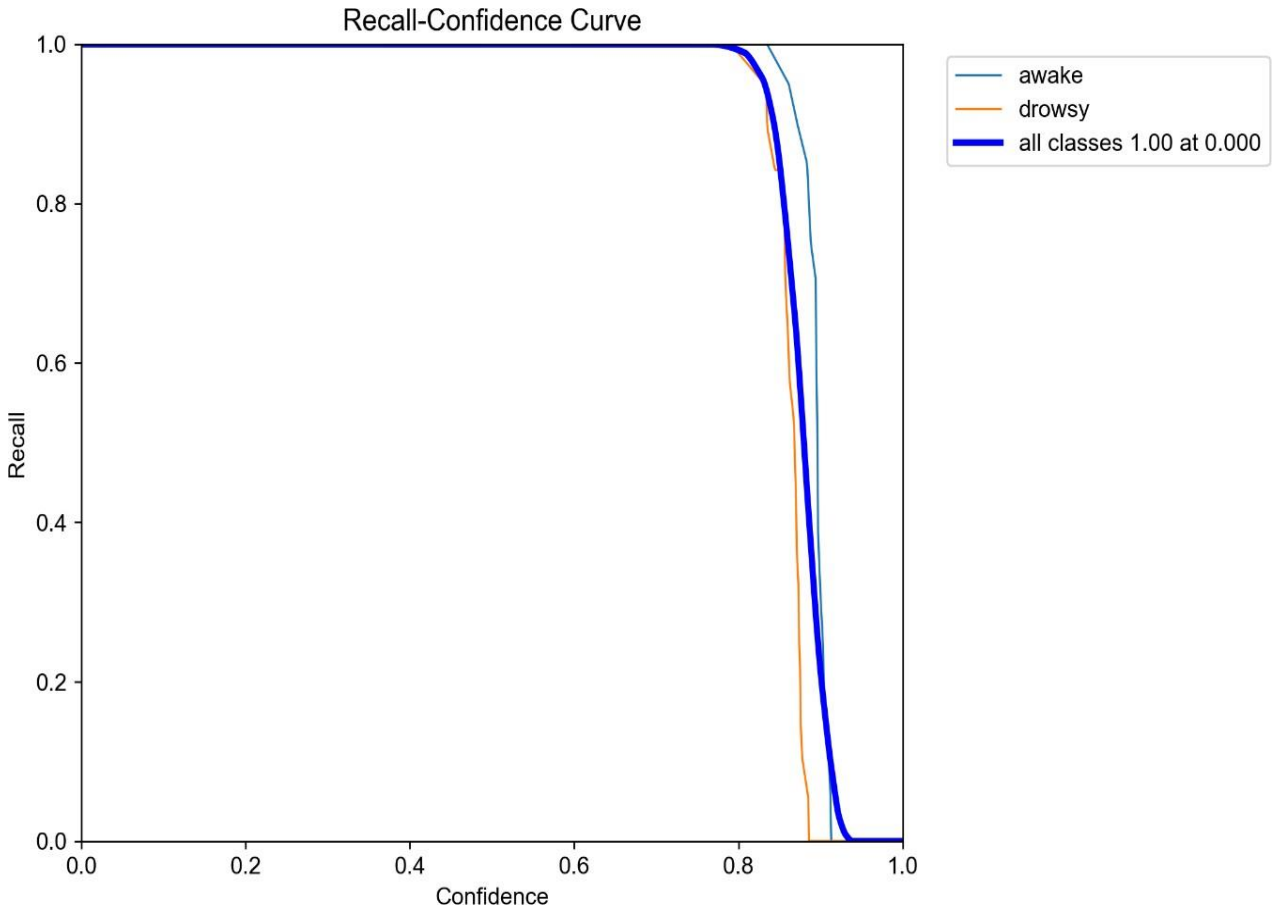


FIG 5.7 R_CURVE

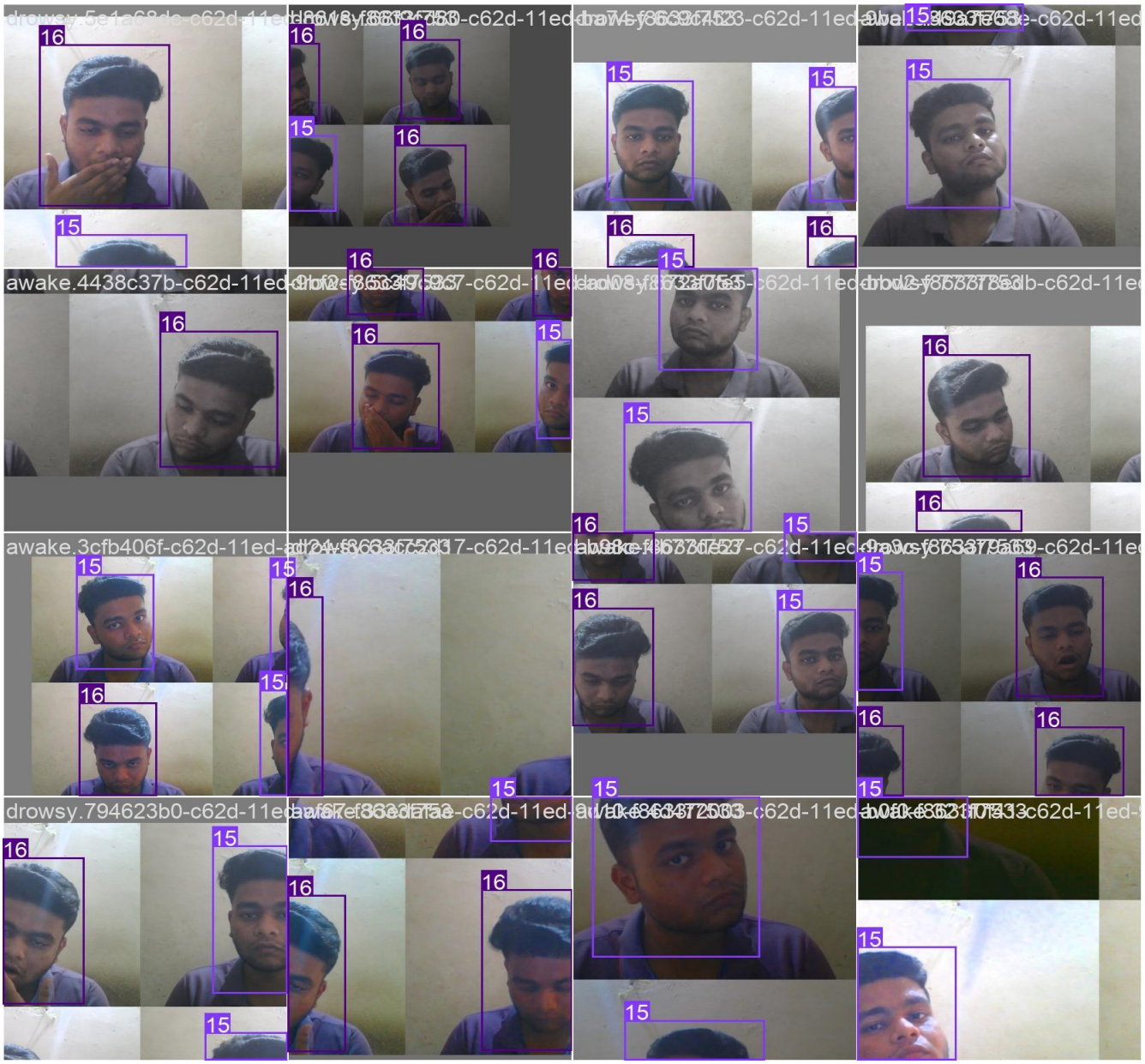


FIG 5.8 TRAIN_BATCH0

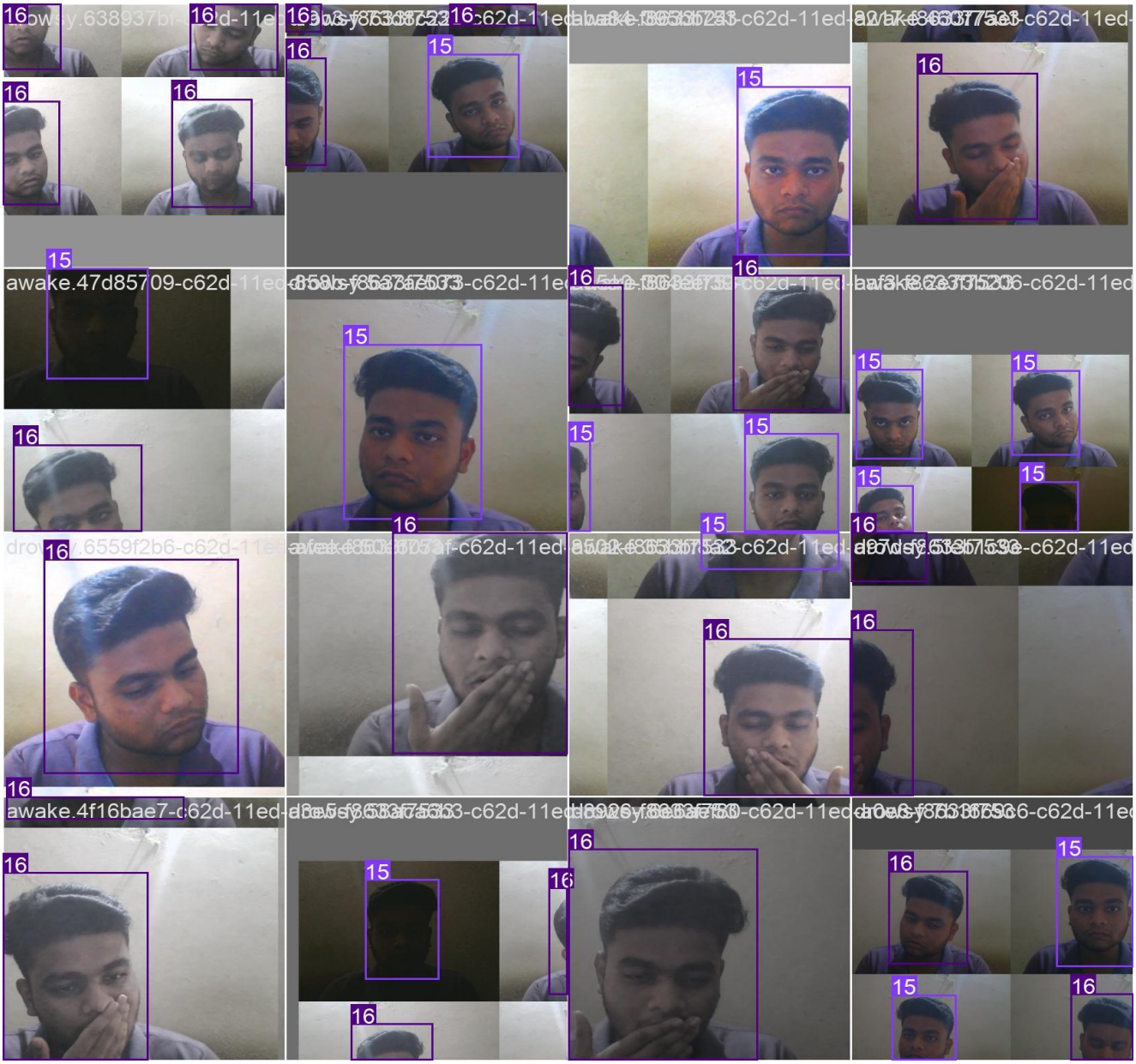


FIG 5.9 TRAIN_BATCH1

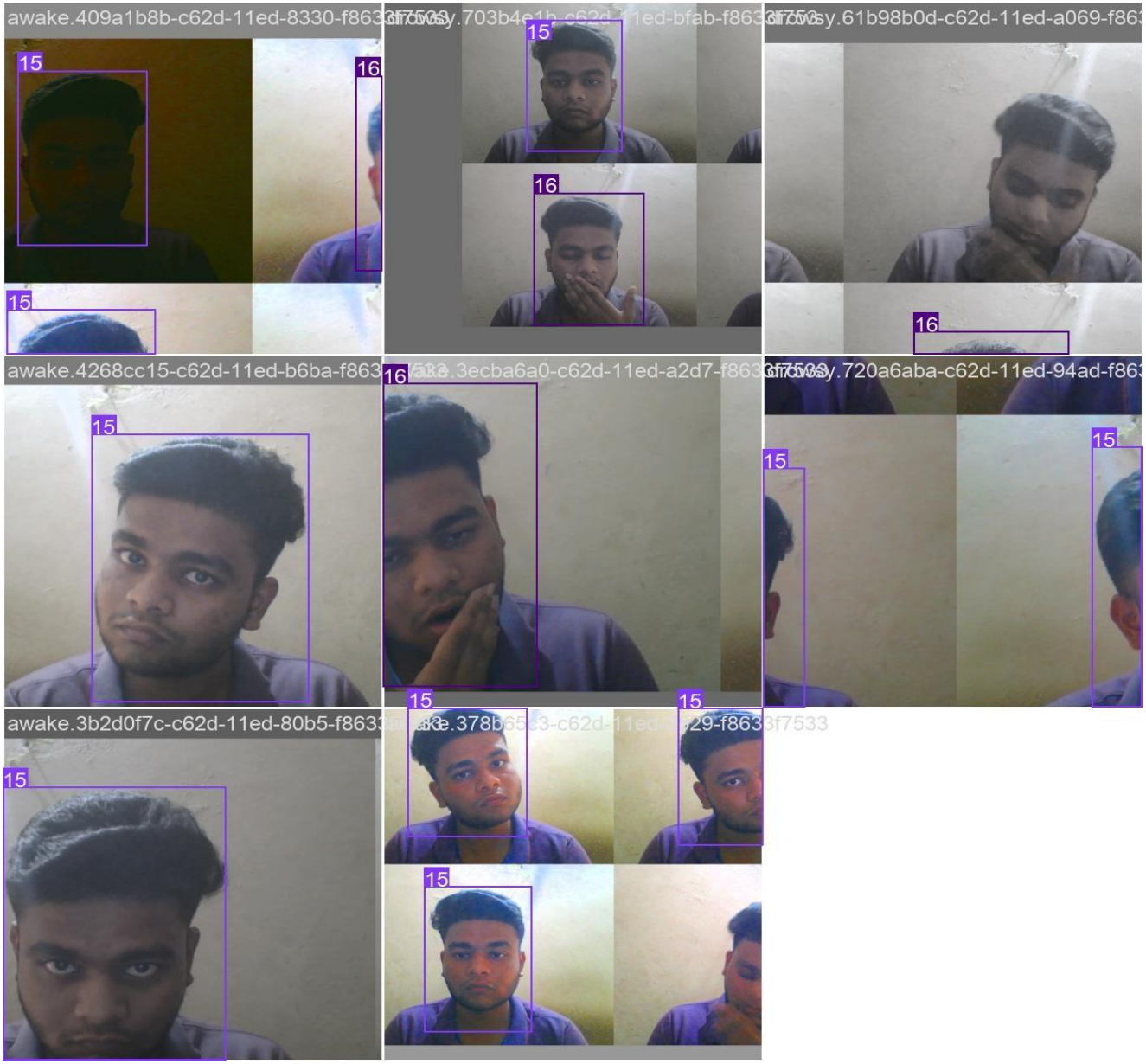


FIG 5.10 TRAIN_BATCH2

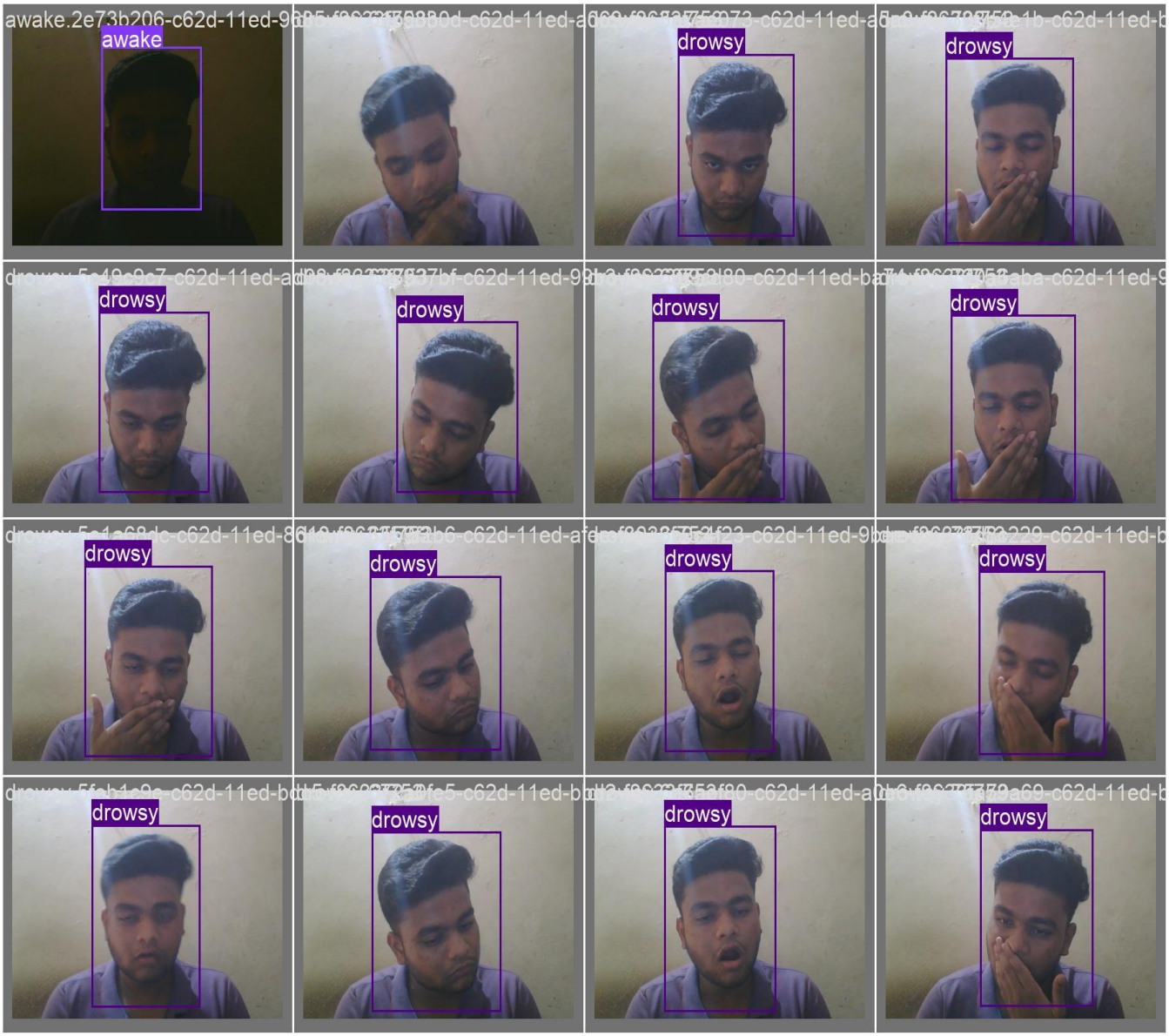


FIG 5.11 VAL_BATCH0_LABELS

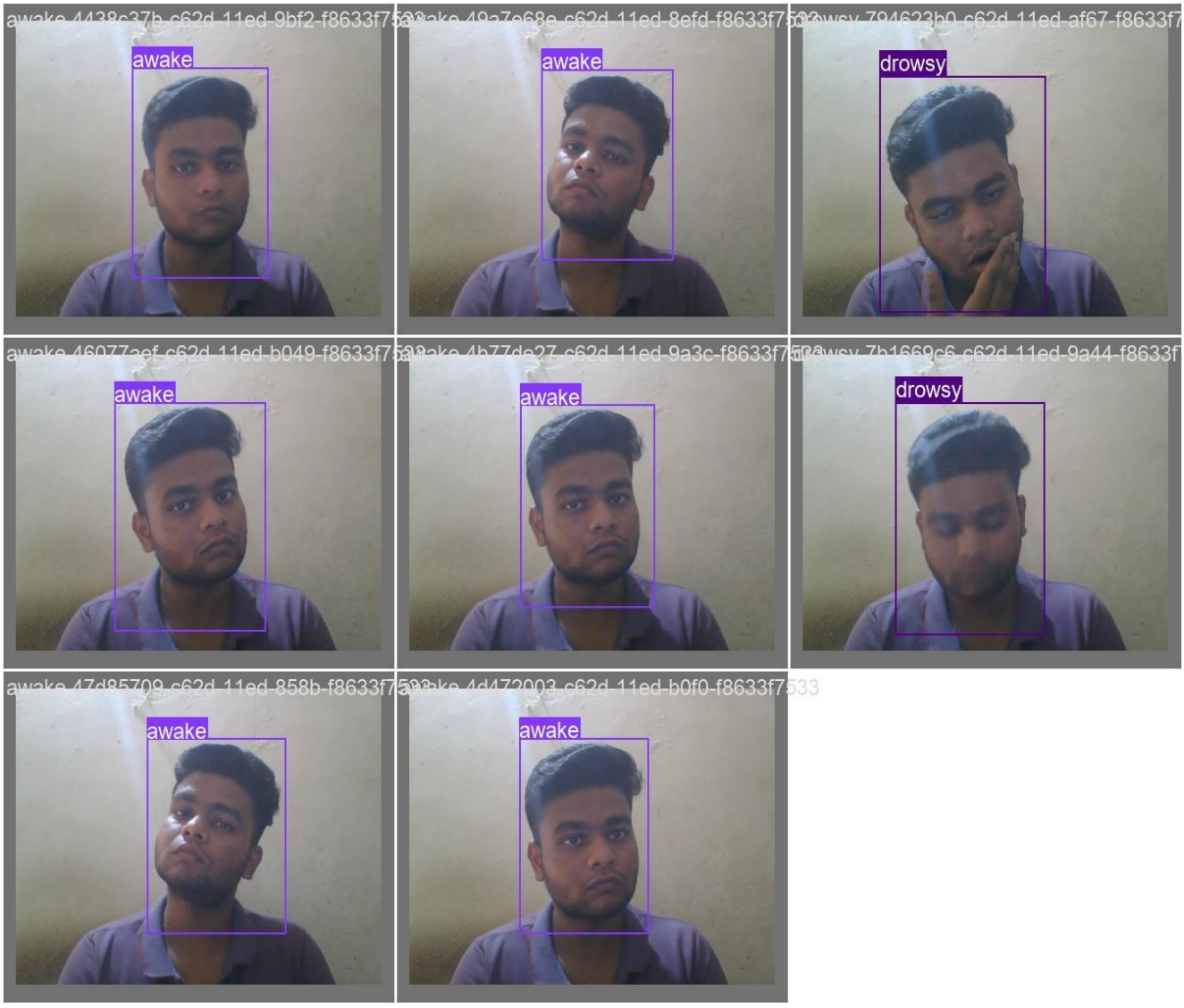


FIG 5.12 VAL_BATCH1_LABELS

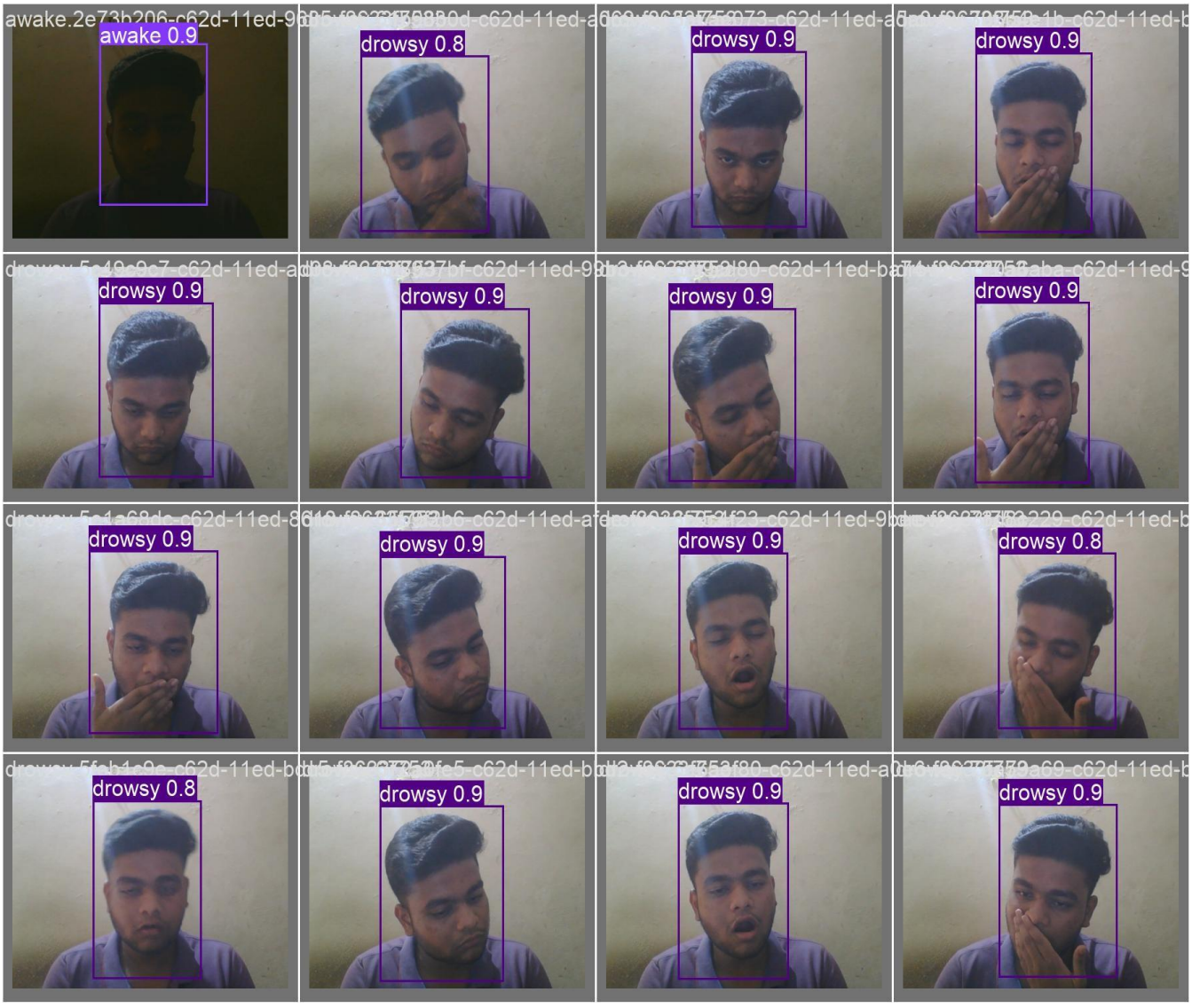


FIG 5.13 VAL_BATCH0_PRED

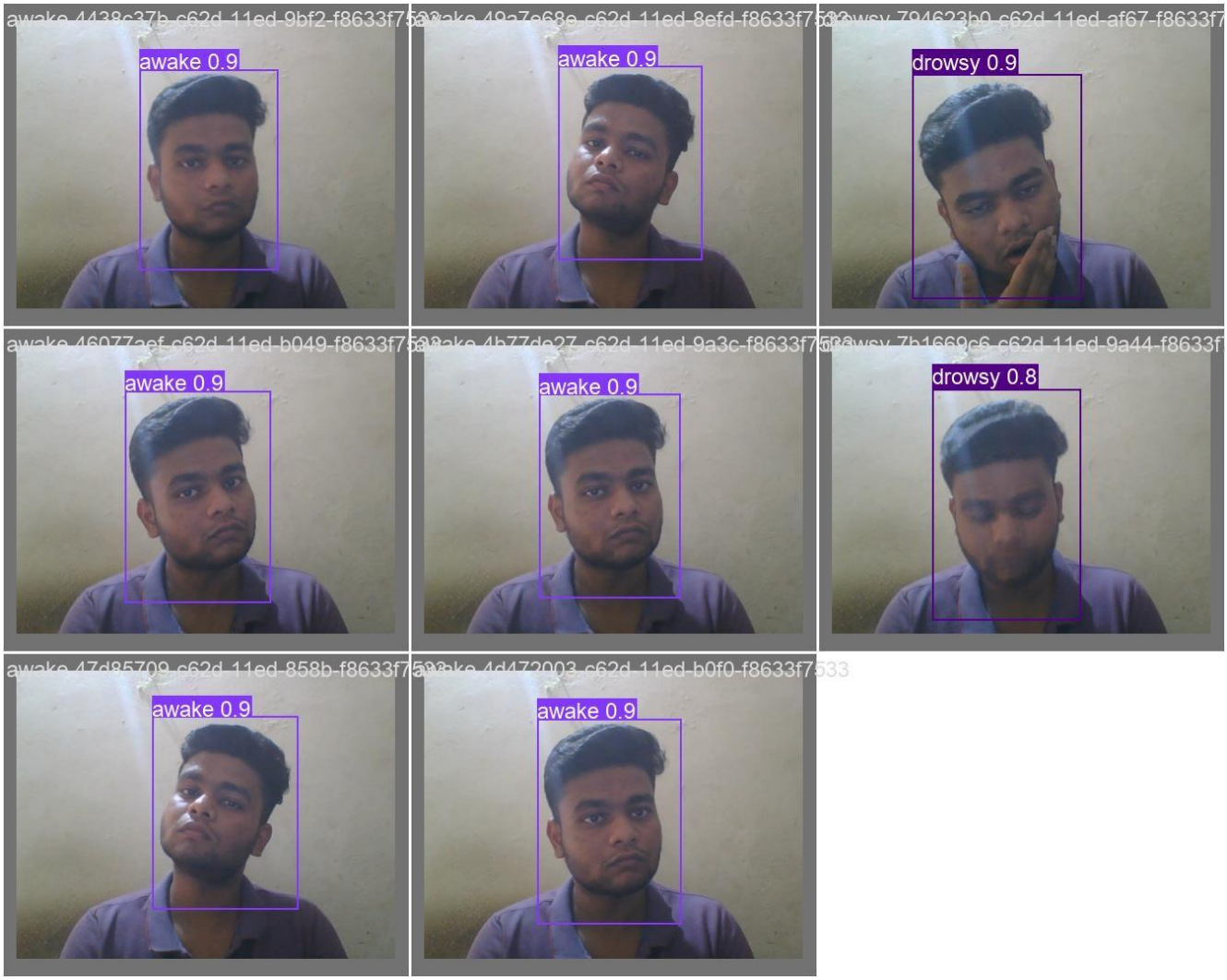


FIG 5.14 VAL_BATCH1_PRED

LIVE DETECTION RESUTS

FOR THE LABEL AWAKE:

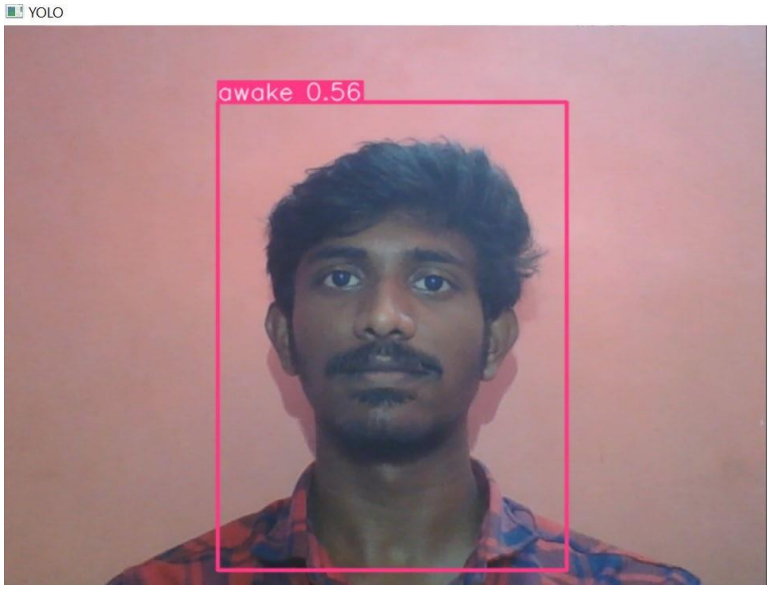


FIG 5.15

FIG 5.16

FOR THE LABEL DROWSY

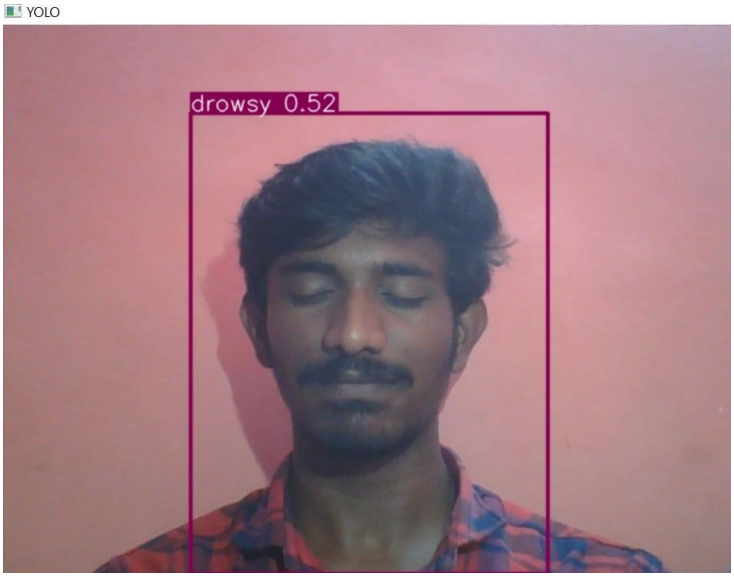


FIG 5.17

FIG 5.18

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 EXPERIMENTAL RESULT

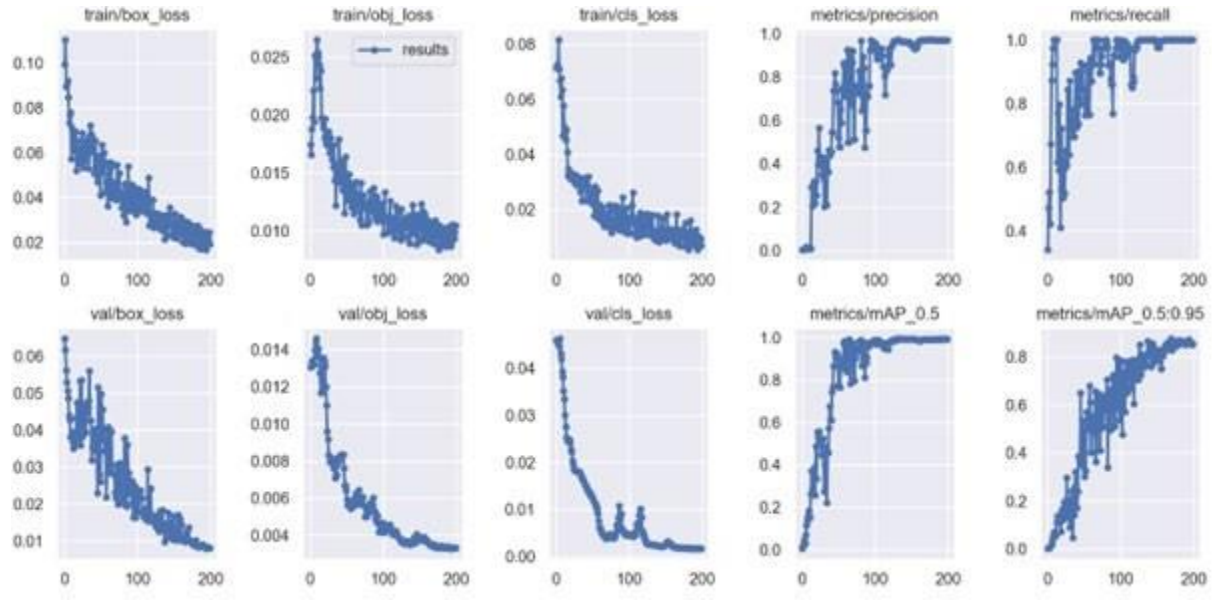


FIG 6.1 RESULTS

6.2 CONCLUSION

In conclusion, this project successfully implemented the YOLOv5 algorithm and LabelImg tool for drowsiness detection using a custom dataset of 40 images, which was trained for 200 epochs. The experimental results showed promising performance in detecting drowsiness, achieving a high mean average precision (mAP) and intersection over union (IoU) for the test dataset.

The YOLOv5 algorithm provided high accuracy in detecting drowsiness and the LabelImg tool enabled the annotation of the custom images for training. The project also involved data preprocessing, augmentation, and splitting to ensure a robust training process.

Through this project, we have demonstrated the effectiveness of deep learning models for drowsiness detection, and the importance of high-quality training data and careful hyperparameter tuning for achieving good performance. Additionally, the use of open-source tools like YOLOv5 and LabelImg has made it possible for researchers and developers to easily experiment with and customize object detection models for their own applications.

Moving forward, there is still much room for improvement in drowsiness detection systems, particularly in challenging environments with varying lighting conditions, camera angles, and user behaviors. Nevertheless, this project represents a valuable contribution to the field of computer vision and has the potential to pave the way for further research in this area.

6.3 FUTURE ENHANCEMENT

The system can be developed in the future by connecting it with mobile framework, allowing it to be made available as an android/iOS application, extending its reach even further. This method can be improved by allowing the user to send himself an alert while driving. This method will also be made available to the general people around the world and be widely used around the universe.

REFERENCE

1. S. Sangle, B. Rathore, R. Rathod, A. Yadav, and A. Yadav, “Real Time Drowsiness Detection System,” pp. 87–92, 2018.
2. V. Varghese, A. Shenoy, S. Ks, and K. P. Remya, “Ear Based Driver Drowsiness Detection System,” vol. 2018, pp. 93–96, 2018.
3. A. Kumar and R. Patra, “Driver drowsiness monitoring system using visual behaviour and machine learning,” ISCAIE 2018 - 2018 IEEE Symposium on Computer Applications and Industrial Electronics, pp. 339–344, 2018.
4. T. Hwang, M. Kim, S. Hong, and K. S. Park, “Driver drowsiness detection using the in-ear EEG,” Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS, vol. 2016–Octob, pp. 4646–4649,2016.
5. S. Junawane, S. Jagtap, P. Deshpande, and L. Soni, “Driver Drowsiness Detection Techniques : A Survey,” vol. 6, no. 11, pp. 2015–2017, 2017.
6. R. Jabbar, K. Al-Khalifa, M. Kharbeche, W. Alhajyaseen, M. Jafari, and S. Jiang, “Real-time Driver Drowsiness Detection for Android Application Using Deep Neural Networks Techniques,” Procedia Computer Science, vol. 130, pp. 400–407, 2018.

APPENDIX

SOURCE CODE

In the file DrowsinessDetection.ipynb [Jupyter Notebook file]

Install and Import Dependencies

```
!pip3 install torch torchvision torchaudio

!git clone https://github.com/ultralytics/yolov5

!cd yolov5 & pip install -r requirements.txt

import torch

from matplotlib import pyplot as plt

import numpy as np

import cv2
```

Load Model

```
model = torch.hub.load('ultralytics/yolov5', 'yolov5s')

model
```

Make Detections with Images

```
import os

img = os.path.join('zidane.jpg')

img

results = model(img)

results.print()
```

```
%matplotlib inline

plt.imshow(np.squeeze(results.render()))

plt.show()
```

Real Time Detections

```
cap = cv2.VideoCapture(0)

while cap.isOpened():

    ret, frame = cap.read()

    # Make detections
    results = model(frame)

    cv2.imshow('YOLO', np.squeeze(results.render()))

    if cv2.waitKey(10) & 0xFF == ord('q'):

        break

cap.release()

cv2.destroyAllWindows()
```

Train from scratch

```
import uuid  # Unique identifier

import os

import time

IMAGES_PATH = os.path.join('data', 'images') #/data/images

labels = ['awake', 'drowsy']

number_imgs = 20

cap = cv2.VideoCapture(0)

# Loop through labels
```

```
for label in labels:

    print('Collecting images for {}'.format(label))

    time.sleep(5)

    # Loop through image range
    for img_num in range(number_imgs):

        print('Collecting images for {}, image number {}'.format(label, img_num))


        # Webcam feed
        ret, frame = cap.read()

        # Naming out image path
        imgname = os.path.join(IMAGES_PATH, label+'.'+str(uuid.uuid1())+'.jpg')

        # Writes out image to file
        cv2.imwrite(imgname, frame)

        # Render to the screen
        cv2.imshow('Image Collection', frame)

        # 2 second delay between captures
        time.sleep(2)

        if cv2.waitKey(10) & 0xFF == ord('q'):

            break

cap.release()

cv2.destroyAllWindows()

!git clone https://github.com/tzutalin/labelImg

!pip install pyqt5 lxml –upgrade

!cd labelImg && pyrcc5 -o libs/resources.py resources.qrc

!cd yolov5 && python train.py --img 320 --batch 16 --epochs 15 --data dataset.yml --weights
yolov5s.pt --workers 2

!cd yolov5 && python train.py --img 320 --batch 16 --epochs 50 --data dataset.yml --weights
yolov5s.pt --workers 2

!cd yolov5 && python train.py --img 320 --batch 16 --epochs 100 --data dataset.yml --weights
yolov5s.pt --workers 2
```

```
#final training
!cd yolov5 && python train.py --img 416 --batch 16 --epochs 200 --data dataset.yml --weights yolov5s.pt --workers 2
```

Load Custom Model

```
model = torch.hub.load('ultralytics/yolov5', 'custom', path='yolov5/runs/train/exp15/weights/last.pt', force_reload=True)

img = os.path.join('data', 'images', 'awake.321f0f41-c62d-11ed-9b71-f8633f7533ab.jpg')

results = model(img)

results.print()

%matplotlib inline

plt.imshow(np.squeeze(results.render()))

plt.show()
```

Final Live Detection

```
cap = cv2.VideoCapture(0)

while cap.isOpened():

    ret, frame = cap.read()

    # Make detections
    results = model(frame)

    cv2.imshow('YOLO', np.squeeze(results.render()))

    if cv2.waitKey(10) & 0xFF == ord('q'):

        break

cap.release()

cv2.destroyAllWindows()
```