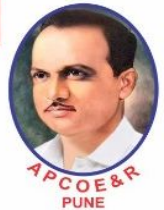
AKHIL BHARTIYA MARATHA SHIKSHAN PARISHAD'S
ANANTRAO PAWAR COLLEGE OF ENGINEERING & RESEARCH
S. No. 103, Parvati, Pune - 411009
Approved by AICTE, New Delhi, Govt. of Maharashtra, Affiliated to Savitribai Phule Pune University
Ph. : 020 24218901/24218959 Email : abmspcoe@yahoo.com Website : www.abmspcoerpune.org
Institute is under Mentorship of College of Engineering, Pune (COEP)
INSTITUTE CODE : EN 6794
Accredited by "NAAC" & ISO Certified



Drowsiness Detection System

Stay Awake, Stay Safe!

Team Members

Siddharth Lanke	B191102019
Pranit Muttha	B191102020
Abhishek Vanshiv	B191102033
Saishubham Laisetti	B191102018

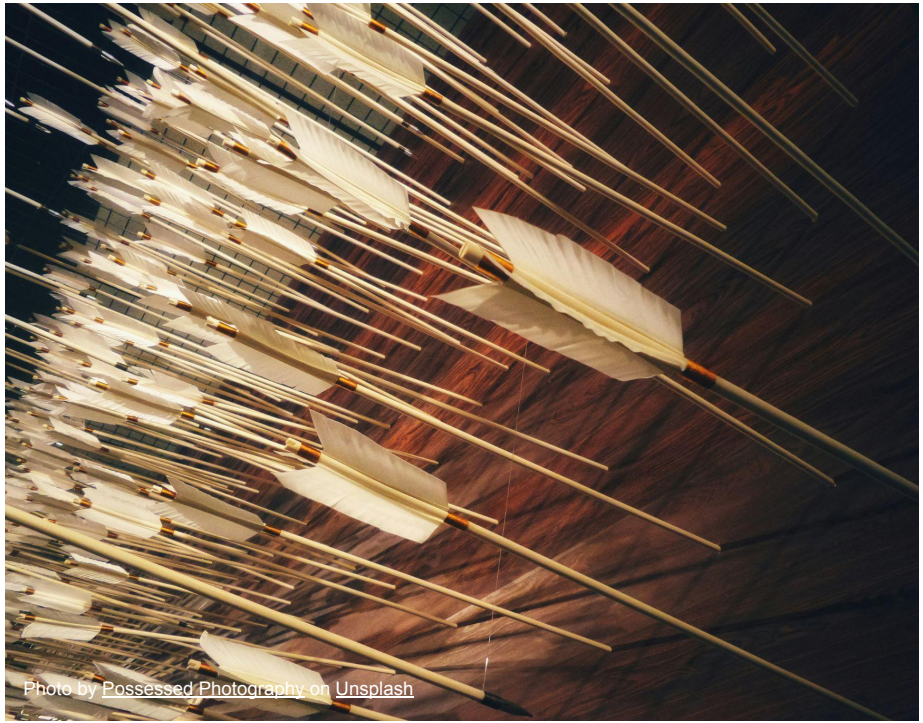
Guide Name: Prof. Swati Joshi

Problem Statement

This project aims to **develop a detection system** using **transfer learning** to identify signs of **drowsiness, fatigue, or similar impairments** in individuals. This system will **promptly alert users** upon detecting these signs, thereby enhancing **safety** and reducing the risk of **accidents or errors** related to impaired alertness.

Objective

- Develop a detection system to identify signs of drowsiness, fatigue, or similar impairments in individuals using transfer learning techniques.
- Utilize pre-trained models and fine-tune them to improve the accuracy of detecting impaired alertness.



Objective

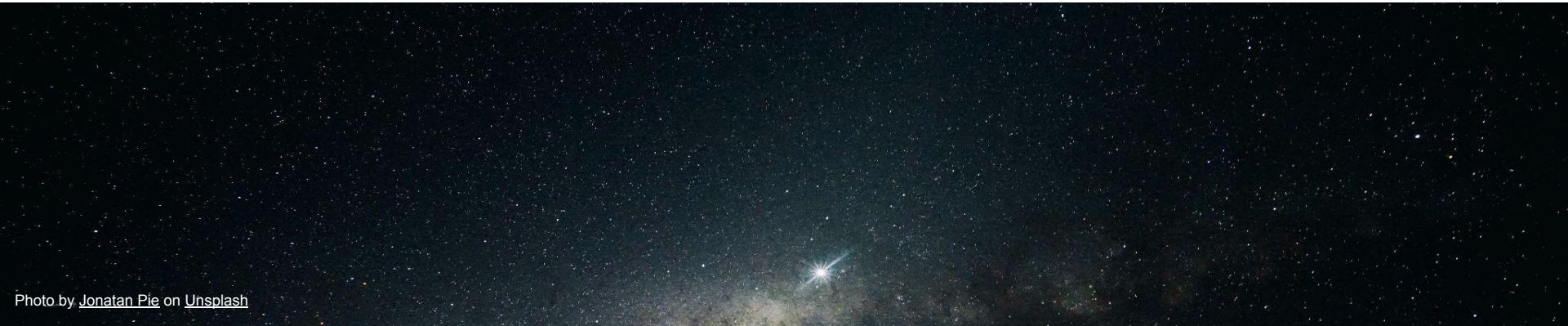
- Implement real-time detection capabilities to ensure timely identification of drowsiness or fatigue.
- Design an alert mechanism to promptly notify users upon detecting signs of impairment.

Software Requirements

- **Google Colab:** Utilized for training the Drowsiness Detection model due to its provision of hardware accelerator using T4 GPU.
- **Ultralytics YOLOv8:** The system utilizes the YOLOv8 architecture implemented by Ultralytics for object detection.
- **Compatible operating systems:** Raspberry Pi OS (previously called Raspbian)

Software Requirements

- **OpenCV:** Required for accessing and processing webcam input, implemented in a local notebook environment.
- **Local Jupyter Notebook/IDE:** For implementing the model in real-time using the laptop webcam for model testing.
- **Roboflow:** Utilized for image annotation and labeling, an essential step in training the Drowsiness Detection model.



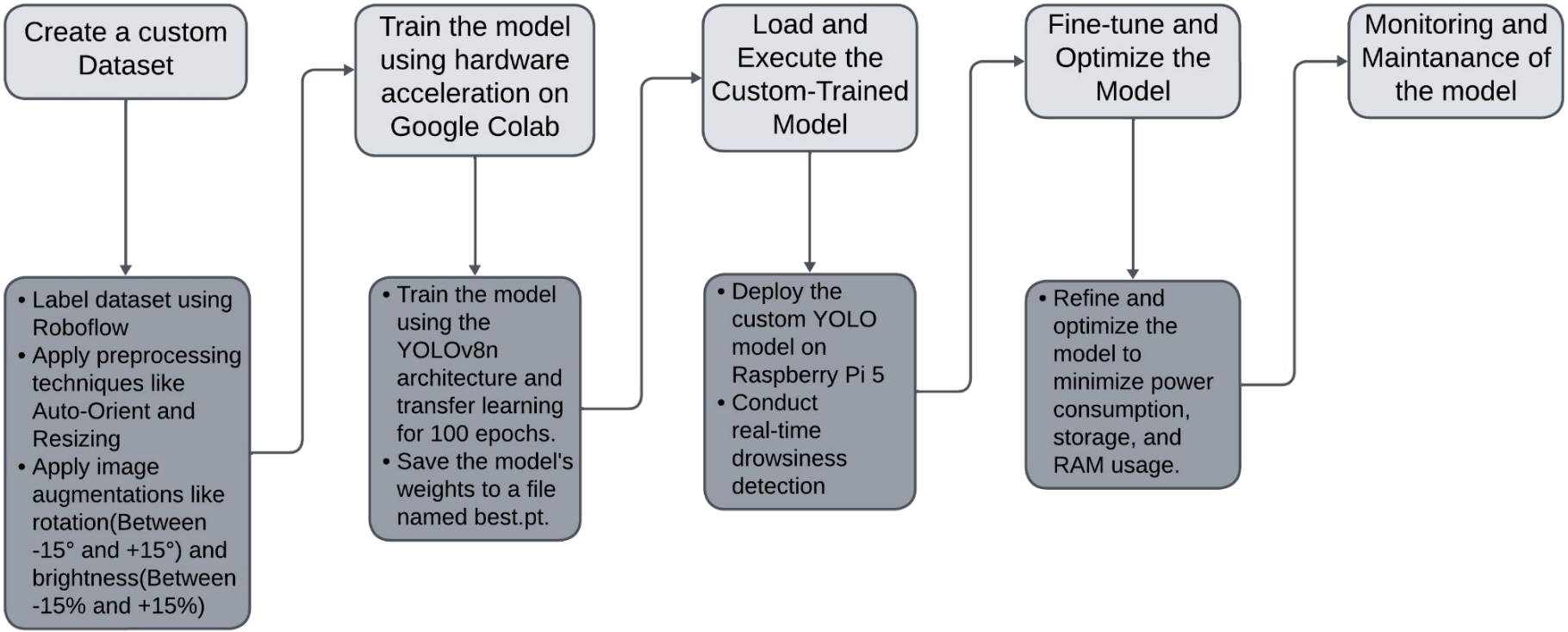
Software Requirements

- **PlayHT:** AI Voice Generator featuring over 600 ultra-realistic Text to Speech voices.

Hardware Requirements

- Raspberry Pi 5 4GB
- Camera Module
- Raspberry Pi 5 Power Supply
- Heatsink
- 32GB Sandisk SD Card
- Mini bluetooth speaker
- Monitor

Project Pipeline

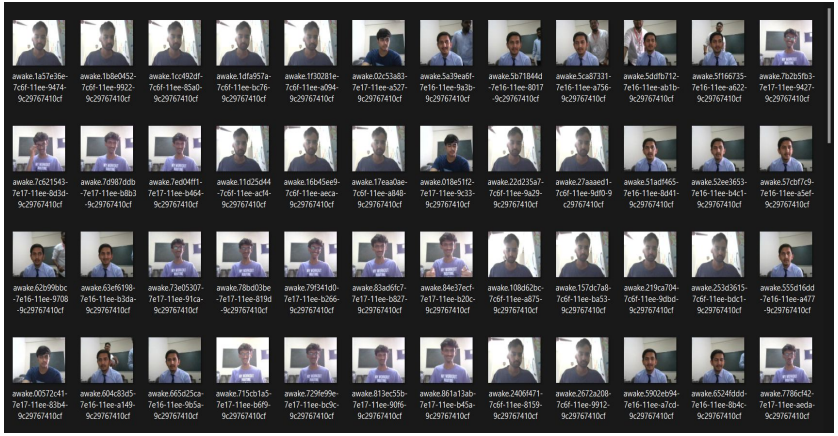


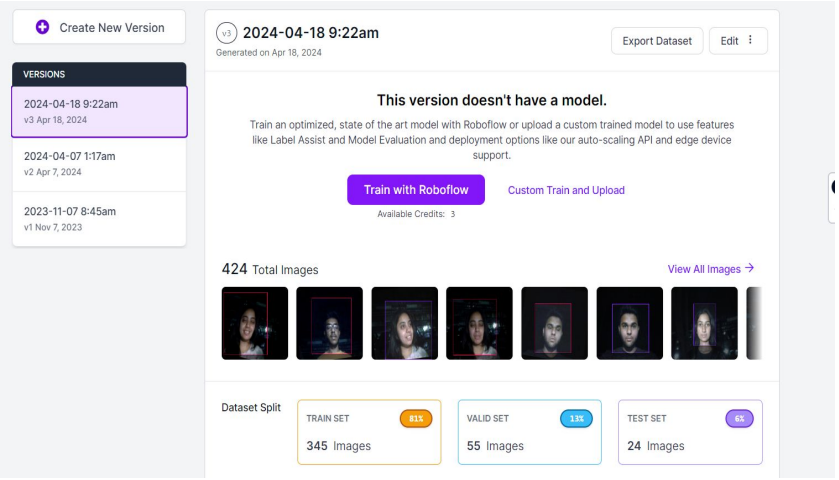
Data Collection

We've put together our own special dataset and are using Roboflow to label and annotate it.

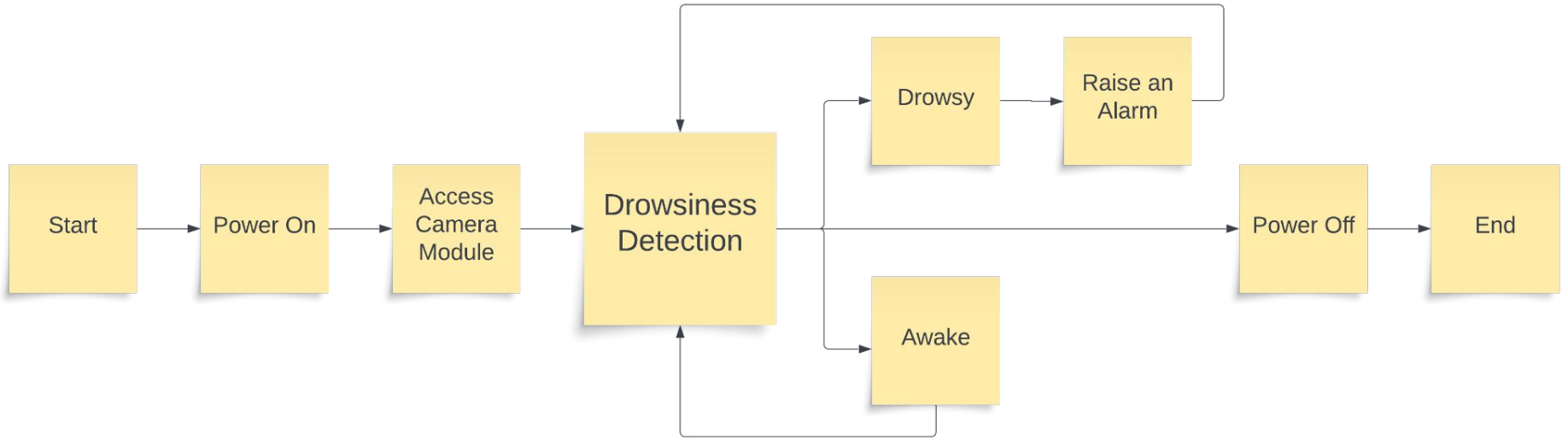
Roboflow helps create high-quality datasets for computer vision projects easily.

We've also looked at publicly available datasets to add more data to our collection.





System Architecture



Code Snippets

Optimized code for Raspberry Pi 5

```python
import cv2
from ultralytics import YOLO
import time
import pygame
import threading

pygame.mixer.init()

audio_files = ['a.mp3', 'b.mp3', 'c.mp3', 'd.mp3', 'e.mp3']
current_audio_index = 0
drowsy_start_time = None
drowsiness_duration = 2

def play_next_audio():
    global current_audio_index
    pygame.mixer.music.load(audio_files[current_audio_index])
    pygame.mixer.music.play()
    current_audio_index = (current_audio_index + 1) % len(audio_files)

def async_play_audio():
    if not pygame.mixer.music.get_busy():
        threading.Thread(target=play_next_audio).start()

model = YOLO('best.pt')

cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

while cap.isOpened():
    success, frame = cap.read()
```

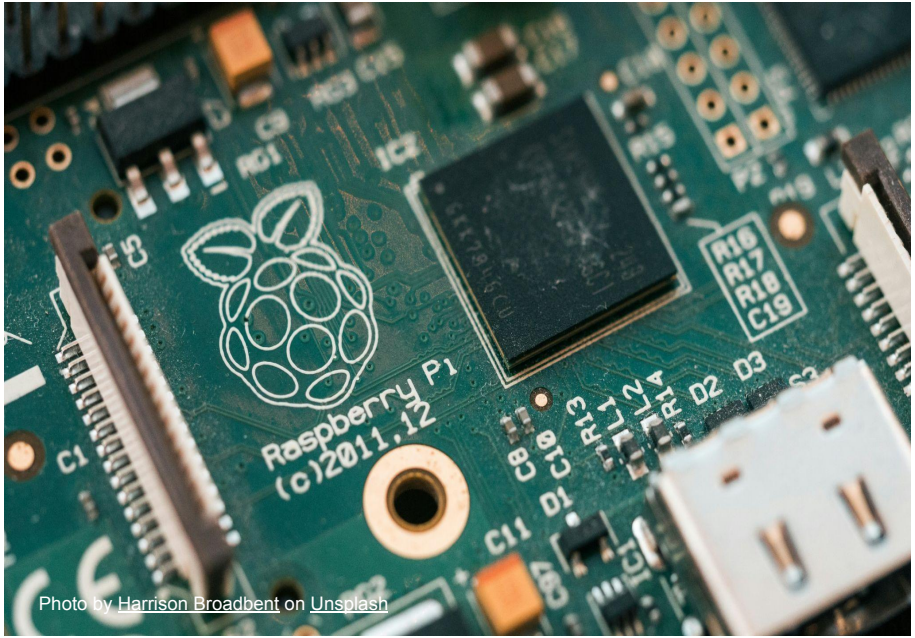
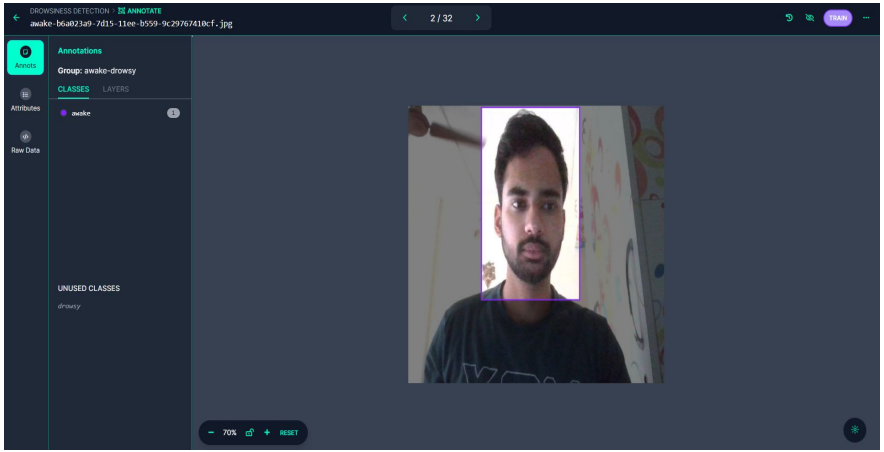


Photo by Harrison Broadbent on Unsplash

Code Snippets: Data Preparation and Training on Google Colab




```
            # Display the annotated frame
            cv2.imshow("YOLOv8 Inference", annotated_frame)

        # Break the loop if 'q' is pressed
        if cv2.waitKey(1) & 0xFF == ord("q"):
            break
    else:
        # Break the loop if the end of the video is reached
        break

# Release the video capture object and close the display window
cap.release()
cv2.destroyAllWindows()
```

```
0: 480x640 1 awake, 20.3ms
Speed: 2.0ms preprocess, 20.3ms inference, 4.0ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 2 awakes, 33.3ms
Speed: 3.1ms preprocess, 33.3ms inference, 6.0ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 1 awake, 1 drowsy, 21.7ms
Speed: 3.4ms preprocess, 21.7ms inference, 3.0ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 1 awake, 23.6ms
Speed: 2.0ms preprocess, 23.6ms inference, 5.0ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 1 awake, 23.7ms
Speed: 3.0ms preprocess, 23.7ms inference, 4.2ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 1 awake, 21.1ms
Speed: 2.0ms preprocess, 21.1ms inference, 3.9ms postprocess per image at shape (1, 3, 480, 640)
```

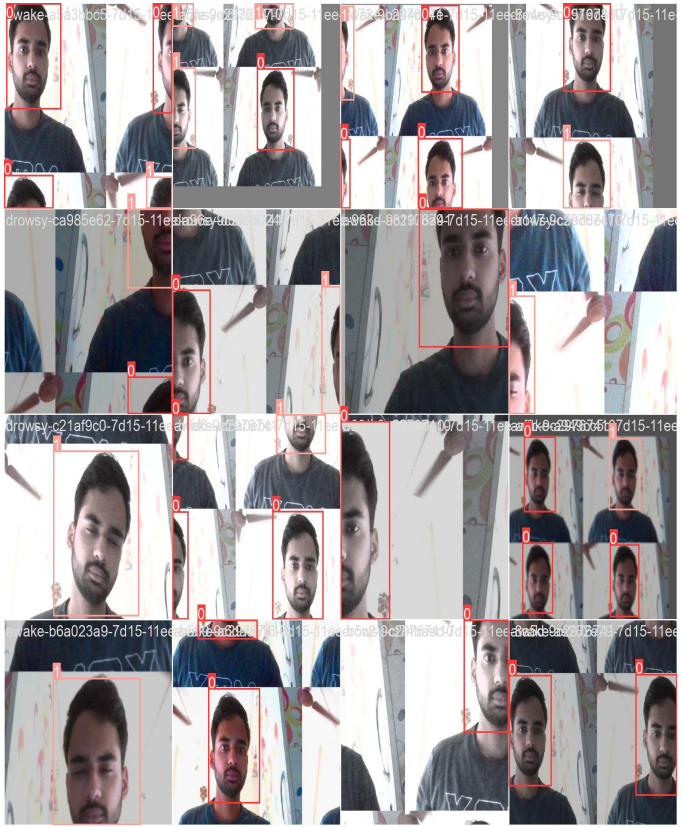
```
Epoch   GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
68/100      0G     0.7696      1.182       1.14         37        640: 100% 2/2 [00:16<00:00,  8.13s/it]
               Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  1.41it/s]
                 all          4          4      0.735       0.75       0.59      0.508

Epoch   GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
69/100      0G     0.8522      1.205       1.151         31        640: 100% 2/2 [00:15<00:00,  8.00s/it]
               Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  1.41it/s]
                 all          4          4      0.735       0.75       0.59      0.508

Epoch   GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
70/100      0G      0.896      1.131       1.182         40        640: 100% 2/2 [00:16<00:00,  8.20s/it]
               Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  1.45it/s]
                 all          4          4      0.808       0.75      0.754      0.652
```

```
[ ] !pip show roboflow

Name: roboflow
Version: 1.1.9
Summary: Official Python package for working with the Roboflow API
Home-page: https://github.com/roboflow-ai/roboflow-python
Author: Roboflow
Author-email: support@roboflow.com
License: UNKNOWN
Location: c:\users\sidla\appdata\local\programs\python\python39\lib\site-packages
Requires: certifi, chardet, cycler, idna, kiwisolver, matplotlib, numpy, opencv-python
Required-by:
```

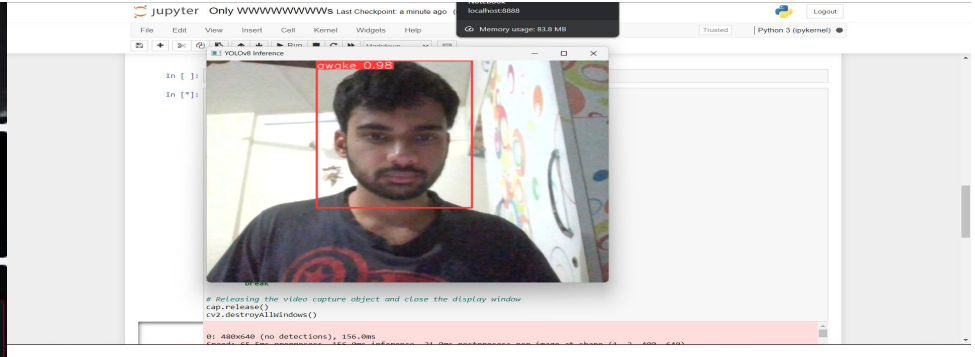
Code Snippets: YOLOv8 Model Initialization and Configuration

```
                 from  n    params  module                                       arguments
  0                -1  1       464  ultralytics.nn.modules.conv.Conv             [3, 16, 3, 2]
  1                -1  1      4672  ultralytics.nn.modules.conv.Conv             [16, 32, 3, 2]
  2                -1  1      7360  ultralytics.nn.modules.block.C2f             [32, 32, 1, True]
  3                -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3, 2]
  4                -1  2     49664  ultralytics.nn.modules.block.C2f             [64, 64, 2, True]
  5                -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
  6                -1  2    197632  ultralytics.nn.modules.block.C2f             [128, 128, 2, True]
  7                -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256, 3, 2]
  8                -1  1    460288  ultralytics.nn.modules.block.C2f             [256, 256, 1, True]
  9                -1  1    164608  ultralytics.nn.modules.block.SPPF            [256, 256, 5]
 10                -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11          [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 12                -1  1    148224  ultralytics.nn.modules.block.C2f             [384, 128, 1]
 13                -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 14          [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 15                -1  1     37248  ultralytics.nn.modules.block.C2f             [192, 64, 1]
 16                -1  1     36992  ultralytics.nn.modules.conv.Conv             [64, 64, 3, 2]
 17         [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 18                -1  1    123648  ultralytics.nn.modules.block.C2f             [192, 128, 1]
 19                -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128, 3, 2]
 20          [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 21                -1  1    493056  ultralytics.nn.modules.block.C2f             [384, 256, 1]
 22        [15, 18, 21]  1    751702  ultralytics.nn.modules.head.Detect           [2, [64, 128, 256]]
Model summary: 225 layers, 3011238 parameters, 3011222 gradients, 8.2 GFLOPs
```

```
[ ] ! pip install ultralytics

Requirement already satisfied: ultralytics in /usr/local/lib/python3.10/dist-packages (8.0.207)
Requirement already satisfied: matplotlib>=3.3.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (3.7.1)
Requirement already satisfied: numpy>=1.22.2 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (1.23.5)
Requirement already satisfied: opencv-python>=4.6.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (4.8.0.76)
Requirement already satisfied: pillow>=7.1.2 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (9.4.0)
Requirement already satisfied: pyyaml>=5.3.1 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (6.0.1)
Requirement already satisfied: requests>=2.23.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (2.31.0)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (1.11.3)
Requirement already satisfied: torch>=1.8.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (2.1.0+cu118)
Requirement already satisfied: torchvision>=0.9.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (0.16.0+cu118)
Requirement already satisfied: tqdm>=4.64.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (4.66.1)
Requirement already satisfied: pandas>=1.1.4 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (1.5.3)
Requirement already satisfied: seaborn>=0.11.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (0.12.2)
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from ultralytics) (5.9.5)
Requirement already satisfied: py-cpuinfo in /usr/local/lib/python3.10/dist-packages (from ultralytics) (9.0.0)
Requirement already satisfied: thop>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (0.1.1.post2209072238)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.3.0->ultralytics) (0.1
```



Code Snippets: Real-time Implementation using OpenCV in a Local Environment




```
# Release the video capture object and close the display window
cap.release()
cv2.destroyAllWindows()
```

```
0: 480x640 1 awake, 148.7ms
Speed: 61.0ms preprocess, 148.7ms inference, 82.5ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 1 drowsy, 140.7ms
Speed: 6.5ms preprocess, 140.7ms inference, 3.0ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 1 drowsy, 29.0ms
Speed: 4.0ms preprocess, 29.0ms inference, 7.0ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 1 awake, 11.0ms
Speed: 2.0ms preprocess, 11.0ms inference, 2.0ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 1 awake, 13.0ms
Speed: 2.0ms preprocess, 13.0ms inference, 3.0ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 1 awake, 1 drowsy, 13.0ms
Speed: 2.0ms preprocess, 13.0ms inference, 3.0ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 1 drowsy, 13.0ms
Speed: 1.0ms preprocess, 13.0ms inference, 3.0ms postprocess per image at shape (1, 3, 480, 640)

0: 480x640 1 drowsy, 15.3ms
Speed: 2.0ms preprocess, 15.3ms inference, 4.0ms postprocess per image at shape (1, 3, 480, 640)
```

Comparison with Traditional Methods:

- Traditional methods lack the robustness and generalization capabilities of deep learning-based approaches like YOLO.
- Performance of traditional methods degrades under challenging conditions such as varying illumination and occlusion.
- Dependency on accurate landmark localization limits traditional methods' effectiveness in real-world applications.

Physiological Signal Monitoring:

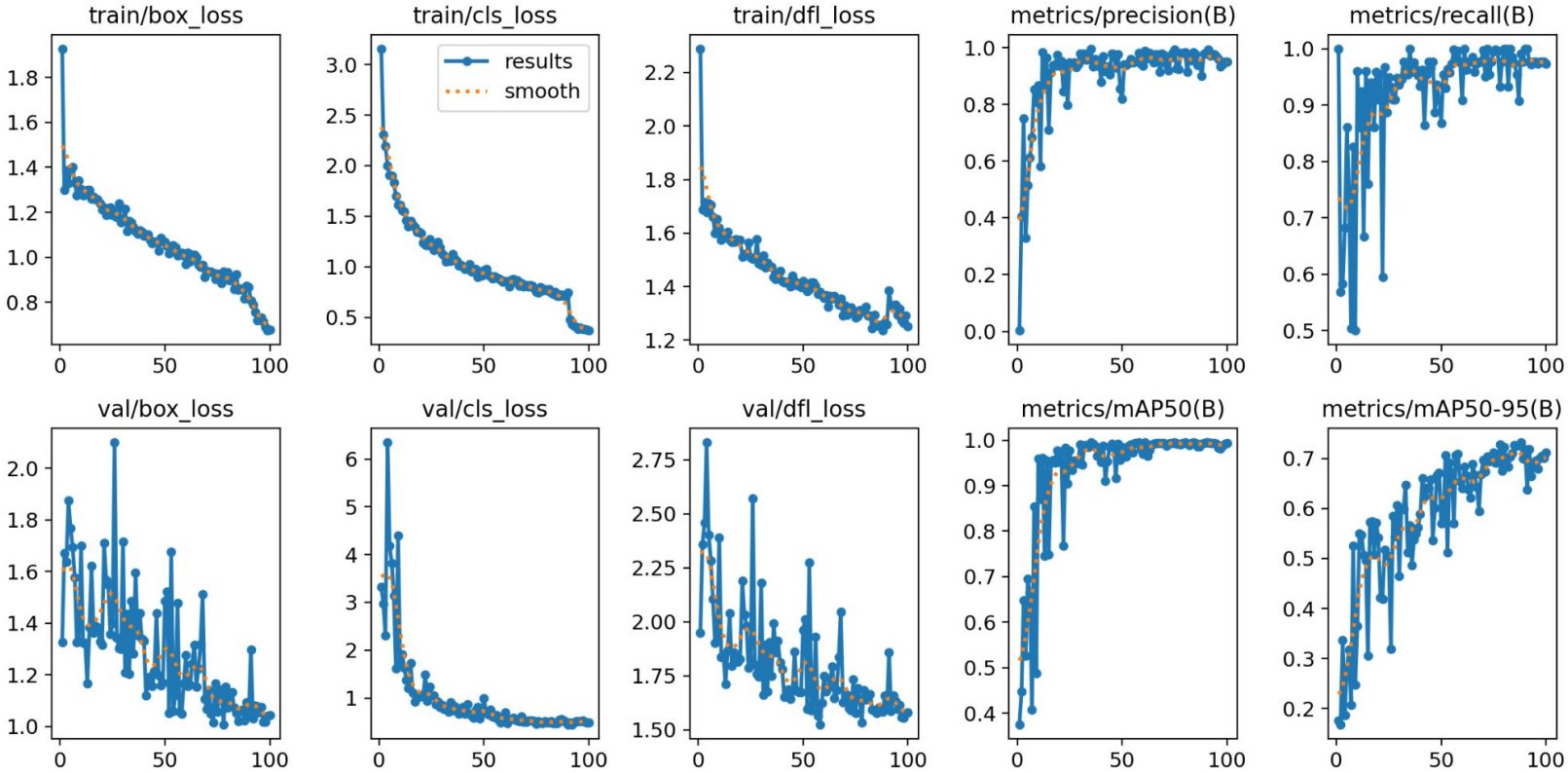
- Monitoring physiological signals like EEG or EOG offers indirect measures of drowsiness but may be limited by complexity and discomfort to users.

Advantages of YOLO-based Drowsiness Detection

- **Utilizes Pre-trained Models:** YOLO uses pre-trained models to detect crucial indicators like eye closure, adaptable to various conditions.
- **Robustness and Adaptability:** YOLO adapts to different lighting and facial orientations, ensuring robust and versatile detection.
- **Automatic Feature Extraction:** Leveraging deep learning, YOLO automatically extracts features for robust detection without manual intervention.
- **Non-invasive and Widely Applicable:** YOLO-based detection is non-invasive, requiring no specialized equipment, making it widely applicable.

Project Implementation Requirements

Project Output



Project Output: Precision and Recall

- **Precision:**
  - metrics/precision(B): High and stable at 0.949, indicating the model's correctness in predicting object presence and location with 94.9% reliability.
- **Recall:**
  - metrics/recall(B): High at 0.973, indicating the model detects 97.3% of all actual positives, showcasing its capacity to find relevant cases.

Summary of Model Accuracy Metrics

- **Precision (Box(P)) at 0.949:** Indicates the model's correctness with a reliability of approximately 94.9%.
- **Recall (R) at 0.973:** Shows the model's ability to detect 97.3% of all actual positives.
- **mAP50 at IoU=50% at 0.993:** Reflects high effectiveness in detecting objects.
- **mAP50-95 at 0.712:** Indicates performance under stricter conditions, with a slight drop from mAP50.

Conclusion

- The YOLO object detection model demonstrates robust learning with decreasing training and validation losses.
- High precision, recall, and mAP metrics indicate the model's reliability and effectiveness in detecting objects across various IoU thresholds.
- Overall, the model performs well in real-world object detection scenarios with minor performance drops under stricter conditions.

Future Scope

- Implementing infrared cameras to enhance low-light performance.
- Utilizing sophisticated algorithms such as reinforcement learning.
- Upgrading to advanced versions of mini-computers like Raspberry Pi for improved performance.
- Training the model on a diverse dataset that includes images of people from all races with unique facial features.

References

- OpenCV - Computer Vision Library: OpenCV Documentation
- PyTorch - Deep Learning Framework: PyTorch Documentation
- Matplotlib - Data Visualization Library: Matplotlib Documentation
- Roboflow - Data Augmentation and Management: Roboflow Documentation
- Ultralytics - YOLOv8: Ultralytics Documentation