**DROWSINESS DETECTOR**

This documentation provides a detailed explanation of two scripts that are used for collecting training data and testing a trained model on a live feed from a CCTV camera. The scripts are:

1. code.py
2. testing.py

The code also allows us to train a yolov5 detection model on out own custom data which we annote using LabelImg.

#### Prerequisites

1. **Python**: Ensure Python is installed on your system.
2. **Libraries**: The following Python libraries are required: torch, opencv-python- headless, matplotlib, numpy, ultralytics.
3. **Hardware**: A webcam is required for capturing images and live feed.
4. LabelImg

### Script 1: code.py

#### Purpose

This script captures images from a webcam and saves them with labels for training a machine learning model. It collects images for two labels: "awake" and "drowsy."

#### Algorithm

1. **Initialize Libraries and Model**:
   * Import necessary libraries for image processing and model handling.
   * Load the YOLO model from the ultralytics library.
2. **Setup Directories and Labels**:
   * Define the directory path for saving images and create the necessary folders if they don't exist.
   * Define labels for the images and the number of images to be captured per label.
3. **Capture Images**:
   * Initialize the webcam for capturing images.
   * Loop through each label and capture the specified number of images:
     + Display a message indicating the label and image number being collected.
     + Capture an image from the webcam.
     + Save the image with a unique filename using the uuid library.
     + Display the captured image in a window.
     + Pause for a short duration before capturing the next image.
     + Allow exiting the loop early by pressing the 'q' key.
4. **Release Resources**:
   * Release the webcam and close all OpenCV windows.

#### Inputs and Outputs

* **Inputs**: None directly; the script uses a webcam to capture images.
* **Outputs**:
  + Captured images saved in the specified directory with filenames indicating their labels.

Annote the the collected images using LableImg. The two classes are supposed to be : “awake” and “drowsy”. Annote only the face of the person as we are detecting drwosiness using facial features. Store the labels in a separate folder.

Run this command to train yolov5 model on the images and labels we annotated:

***python train.py --img 320 --batch 16 --epochs 500 --data dataset.yml --weights yolov5s.pt --workers 2***

### Script 2: testing.py

#### Purpose

This script uses a trained YOLO model to perform real-time object detection on a live feed from a webcam. It displays the detection results in a window.

#### Algorithm

1. **Initialize Libraries and Model**:
   * Import necessary libraries for image processing and model handling.
   * Load the custom-trained YOLO model from the specified path.
2. **Setup Webcam**:
   * Initialize the webcam for capturing the live feed.
3. **Real-Time Detection**:
   * Loop while the webcam is open:
     + Capture a frame from the webcam.
     + Perform object detection on the captured frame using the YOLO model.
     + Render the detection results on the frame.
     + Display the frame with the detection results in a window.
     + Allow exiting the loop early by pressing the 'q' key.
4. **Release Resources**:
   * Release the webcam and close all OpenCV windows.

#### Inputs and Outputs

* **Inputs**: None directly; the script uses a webcam to capture the live feed.
* **Outputs**:
  + Real-time display of the webcam feed with object detection results overlaid.

### Workflow Summary

1. **Run** code.py: Collect labeled images for training the model.
2. **Train the YOLO Model**: Use the collected images to train a custom YOLO model. This step is assumed to be done separately and the trained model is saved.
3. **Run** testing.py: Use the trained YOLO model to perform real-time object detection on the live feed from a webcam.

By following this workflow, you can collect data for training a model and then test the trained model on a live feed to verify its performance.

The code and the workflow is inspired from the below videos:

Link :https://youtu.be/tFNJGim3FXw?si=rPx7kUtXZQP2u8Bk