UNIVERSITY OF ENGINEERING AND TECHNOLOGY, TAXILA

COMPUTER ENGINEERING DEPARTMENT

Project Report

For the Course Of

Machine Learning (ML)

For

Sixth Semester

Lab:	**Machine Learning (ML)**
INSTRUCTOR:	**Engr. Shahid Ali Bhutta**
Submitted by:	**Muhammad Omar Farooq, Muhammad Zia Ullah, Umair Aziz**
Reg. number:	**20-CP-33, 20-CP-43, 20-CP-47**
Submission date:	**July 16, 2023 (Sun)**

Table of Contents

1. **Problem Statement:** 2
2. **Aims/ Objectives:** 2
3. **Abstract:** 2
4. **Introduction:** 2
5. **Methodology:** 3
6. **CODE:** 4
7. **GUI View:** 8
8. **Conclusion:** 8

Driver Drowsiness Detection System

1. Problem Statement:

The problem addressed in this project is driver drowsiness while operating a vehicle. Drowsy driving can lead to accidents, injuries, and fatalities. The objective is to develop a system that can detect driver drowsiness in real-time and provide an alert to prevent potential accidents.

2. Aims/Objectives:

The main objective of this project is to develop a driver drowsiness detection system using computer vision and deep learning techniques. The system should be able to:

- Detect faces and eyes in real-time using a webcam.
- Classify the status of the driver's eyes as 'Open' or 'Closed'.
- Calculate a score based on the duration of closed eyes.
- Provide an alert when the score exceeds a threshold indicating drowsiness.

3. Abstract:

This project presents a driver drowsiness detection system that utilizes computer vision and deep learning algorithms. The system captures images from a webcam, detects faces and eyes, and feeds them into a trained convolutional neural network (CNN) model. The model classifies the eye status as 'Open' or 'Closed', and a score is calculated based on the duration of closed eyes. When the score exceeds a threshold, an alarm sound is played to alert the driver.

4. Introduction:

The introduction section provides an overview of the problem of drowsy driving and its potential consequences. It highlights the importance of developing a real-time drowsiness detection system to ensure driver safety. The section also explains the motivation behind using computer vision and deep learning techniques for this project.

5. Methodology:

This section describes the methodology used to build the driver drowsiness detection system. It explains the step-by-step process involved in capturing images, detecting faces and eyes, and classifying the eye status using a pre-trained CNN model. The scoring mechanism and alarm system are also discussed in detail.

Let’s understand how our algorithm works step by step.

Step 1 – Take Image as Input from a Camera

With a webcam, we will take images as input. So, to access the webcam, we made an infinite loop that will capture each frame. We use the method provided by OpenCV, *cv2.VideoCapture(0)* to access the camera and set the capture object (cap). *cap.read()* will read each frame and we store the image in a frame variable.

Step 2 – Detect Face in the Image and Create a Region of Interest (ROI)

To detect the face in the image, we need to first convert the image into grayscale as the OpenCV algorithm for object detection takes gray images in the input. We don’t need color information to detect the objects. We will be using haar cascade classifier to detect faces. This line is used to set our classifier.

face = cv2.CascadeClassifier(‘path to our haar cascade xml file’).

Then we perform the detection using

faces = face.detectMultiScale(gray).

It returns an array of detections with x,y coordinates, and height, the width of the boundary box of the object. Now we can iterate over the faces and draw boundary boxes for each face.

for (x,y,w,h) in faces:
cv2.rectangle(frame, (x,y), (x+w, y+h), (100,100,100), 1)

Step 3 – Detect the eyes from ROI and feed it to the classifier.

The same procedure to detect faces is used to detect eyes. First, we set the cascade classifier for eyes in **leye** and **reye** respectively then detect the eyes using *left_eye = leye.detectMultiScale(gray).* Now we need to extract only the eyes data from the full image. This can be achieved by extracting the boundary box of the eye and then we can pull out the eye image from the frame with this code.

l_eye = frame[y : y+h, x : x+w]

l_eye only contains the image data of the eye. This will be fed into our CNN classifier which will predict if eyes are open or closed. Similarly, we will be extracting the right eye into **r_eye**.

Step 4 – Classifier will Categorize whether Eyes are Open or Closed

We are using CNN classifier for predicting the eye status. To feed our image into the model, we need to perform certain operations because the model needs the correct dimensions to start with. First, we convert the color image into grayscale using *r_eye = cv2.cvtColor(r_eye, cv2.COLOR_BGR2GRAY).*

Then, we resize the image to 24*24 pixels as our model was trained on 24*24 pixel images *cv2.resize(r_eye, (24,24)).* We normalize our data for better convergence *r_eye = r_eye/255* (All values will be between 0-1). Expand the dimensions to feed into our classifier.

We loaded our model using *model = load_model(‘models/cnnCat2.h5’)*. Now we predict each eye with our model *lpred = model.predict_classes(l_eye)*. If the value of lpred[0] = 1, it states that eyes are open, if value of lpred[0] = 0 then, it states that eyes are closed.

Step 5 – Calculate Score to Check whether Person is Drowsy

The score is basically a value we will use to determine how long the person has closed his eyes. So if both eyes are closed, we will keep on increasing score and when eyes are open, we decrease the score. We are drawing the result on the screen using cv2.putText() function which will display real time status of the person.

cv2.putText(frame, “Open”, (10, height-20), font, 1, (255,255,255), 1, cv2.LINE_AA)

A threshold is defined for example if score becomes greater than 15 that means the person's eyes are closed for a long period of time. This is when we beep the alarm using **sound.play()**

6. Code:

The code section presents the Python code for the driver drowsiness detection system. It includes the necessary libraries, model loading, webcam access, image processing, classification, scoring, and alarm functionalities. The code is properly commented to enhance understanding and readability. Following is the code:

```
import tkinter as tk
import cv2
import os
from keras.models import load_model
import numpy as np
from pygame import mixer

mixer.init()
sound = mixer.Sound('alarm.wav')

face = cv2.CascadeClassifier('haar cascade files/haarcascade_frontalface_alt.xml')
leye = cv2.CascadeClassifier('haar cascade files/haarcascade_lefteye_2splits.xml')
reye = cv2.CascadeClassifier('haar cascade files/haarcascade_righteye_2splits.xml')

lbl = ['Close', 'Open']

model = load_model('models/cnncat2.h5')
path = os.getcwd()

score = 0
rpred = [99]
lpred = [99]
capture_video = False


def detect_blink():
  global score, rpred, lpred, capture_video

  cap = cv2.VideoCapture(0)
  font = cv2.FONT_HERSHEY_COMPLEX_SMALL

  # Define the VideoWriter object
  fourcc = cv2.VideoWriter_fourcc(*'mp4v')
  out = cv2.VideoWriter('output.mp4', fourcc, 20.0, (640, 480))

  while True:
```

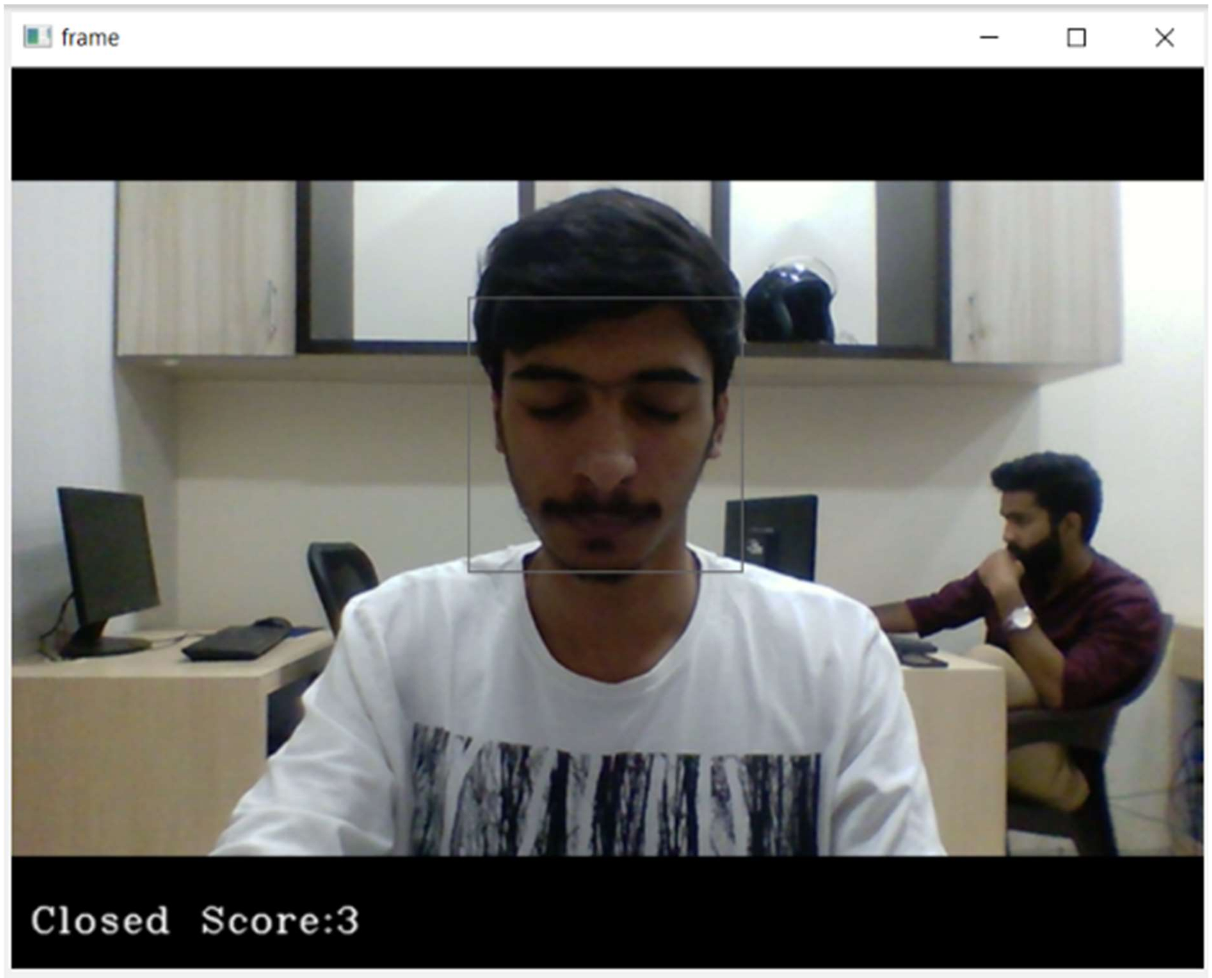
```
ret, frame = cap.read()
height, width = frame.shape[:2]

gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

faces = face.detectMultiScale(gray, minNeighbors=5, scaleFactor=1.1, minSize=(25, 25))
left_eye = leye.detectMultiScale(gray)
right_eye = reye.detectMultiScale(gray)

cv2.rectangle(frame, (0, height - 50), (200, height), (0, 0, 0), thickness=cv2.FILLED)

for (x, y, w, h) in faces:
    cv2.rectangle(frame, (x, y), (x + w, y + h), (100, 100, 100), 1)

for (x, y, w, h) in right_eye:
    r_eye = frame[y:y + h, x:x + w]
    r_eye = cv2.cvtColor(r_eye, cv2.COLOR_BGR2GRAY)
    r_eye = cv2.resize(r_eye, (24, 24))
    r_eye = r_eye / 255
    r_eye = r_eye.reshape(24, 24, -1)
    r_eye = np.expand_dims(r_eye, axis=0)
    rpred = np.argmax(model.predict(r_eye), axis=-1)
    if rpred[0] == 1:
        lbl = 'Open'
    if rpred[0] == 0:
        lbl = 'Closed'
    break

for (x, y, w, h) in left_eye:
    l_eye = frame[y:y + h, x:x + w]
    l_eye = cv2.cvtColor(l_eye, cv2.COLOR_BGR2GRAY)
    l_eye = cv2.resize(l_eye, (24, 24))
    l_eye = l_eye / 255
    l_eye = l_eye.reshape(24, 24, -1)
    l_eye = np.expand_dims(l_eye, axis=0)
    lpred = np.argmax(model.predict(l_eye), axis=-1)
    if lpred[0] == 1:
        lbl = 'Open'

    if lpred[0] == 0:
        lbl = 'Closed'
    break

if rpred[0] == 0 and lpred[0] == 0:
    score = score + 1
else:
```

```
            score = score - 1

        if score < 0:
            score = 0

        score_text = f'Score: {score}'
        cv2.putText(frame, lbl, (10, height - 20), font, 1, (255, 255, 255), 1, cv2.LINE_AA)
        cv2.putText(frame, score_text, (10, height - 5), font, 1, (255, 255, 255), 1, cv2.LINE_AA)

        if score > 15:
            # person is feeling sleepy so we beep the alarm
            cv2.imwrite(os.path.join(path, 'image.jpg'), frame)
            try:
                sound.play()
            except:
                pass

        cv2.imshow('frame', frame)

        if capture_video:
            out.write(frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    # Release the VideoWriter object
    out.release()

    cap.release()
    cv2.destroyAllWindows()


def toggle_video_capture():
    global capture_video
    capture_video = not capture_video
    if capture_video:
        capture_button.config(text='Stop Capture')
    else:
        capture_button.config(text='Start Capture')


# Create the GUI window
window = tk.Tk()
window.title('Drowsiness Detection')
window.geometry('400x250')
```

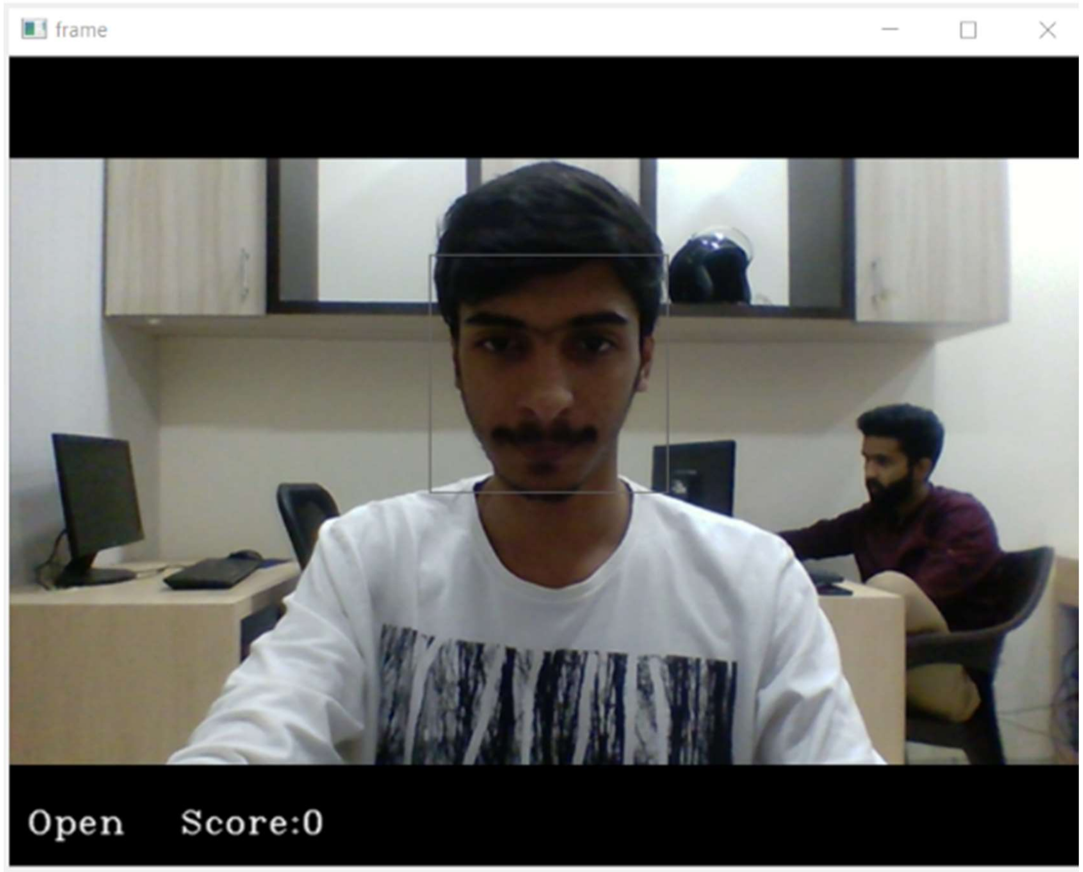
```
# Create a button to start/stop the eye blink detection
start_button = tk.Button(window, text='Start Detection', command=detect_blink)
start_button.pack(pady=10)

# Create a button to start/stop video capture
capture_button = tk.Button(window, text='Start Capture', command=toggle_video_capture)
capture_button.pack(pady=10)

window.mainloop()
```

7. GUI View:

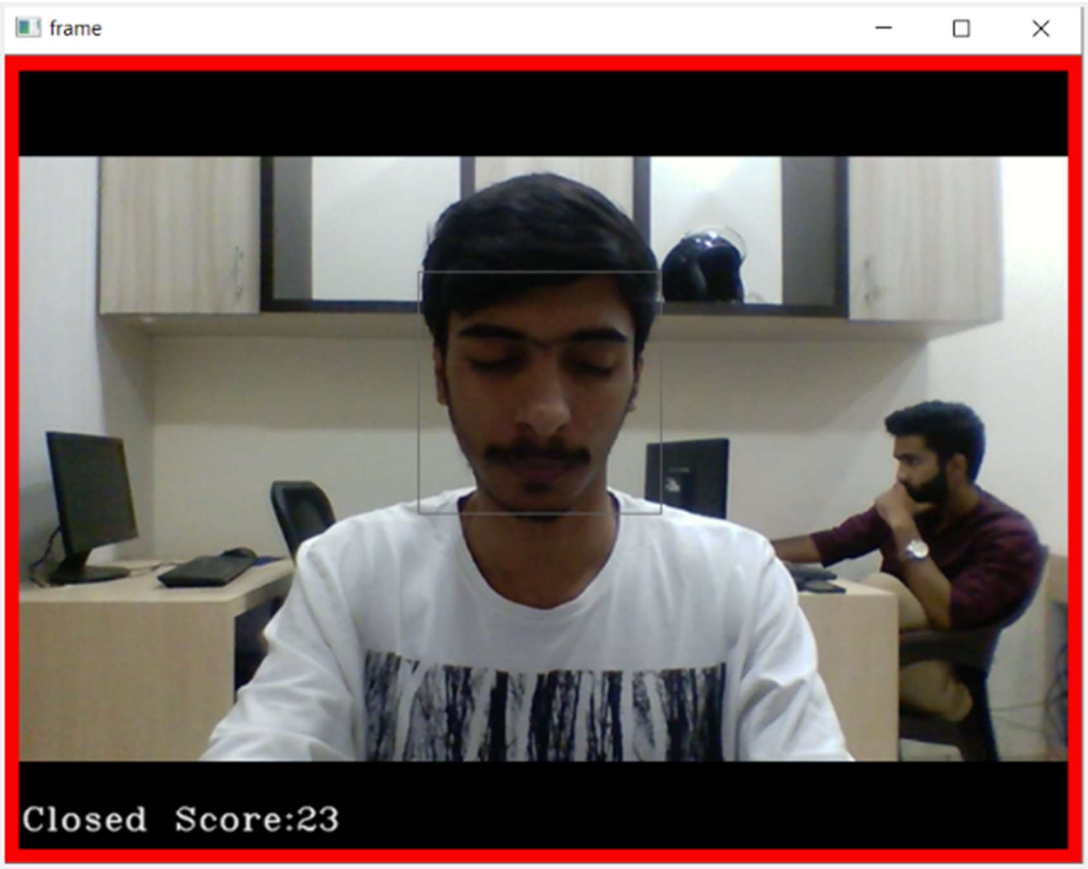
This section shows the graphical user interface (GUI) view of the driver drowsiness detection system.



Closed Eye Detection



Open Eyes Detection



Sleep Alert

8. Conclusion:

The conclusion summarizes the project's objectives, methodology, and outcomes. It emphasizes the effectiveness of the driver drowsiness detection system in real-time scenarios and highlights its potential to enhance driver safety. Future improvements and possible extensions of the system can also be achieved using higher modifications in the model.

****