**Drowsiness Detection System**

The project aims to aid the prevention of accidents.

WHAT

1. It is a computer vision application which beeps the alarm if the person is drowsy
2. This application can get the image from different sources such as video, webcam, IP Camera
3. This application will be used using a browser on a Local Area Network.

WHY

1. This can be used in vehicles to notify the driver if he feels too drowsy.

HOW

Backend: Code, Functionalities

Face and eye Detection: OpenCV

Drowsiness Detection: Tensorflow

Frontend: Interface

Web Application: Streamlit

1.Overview and OpenCV

2.Face and left eye and right eye Detection

3.Drowsiness Detection

4.Frontend Development

5.Feature Integration

Cnn.py

from keras.layers import Conv2D,MaxPooling2D,Flatten,Dense

from keras.models import Sequential

from keras.preprocessing.image import ImageDataGenerator #To organize the data properly

#Define the model

mymodel=Sequential()

#To add Layer

mymodel.add(Conv2D(32,(3,3),activation='relu',input\_shape=(150,150,3)))

mymodel.add(MaxPooling2D())

mymodel.add(Conv2D(32,(3,3),activation='relu'))

mymodel.add(MaxPooling2D())

mymodel.add(Conv2D(32,(3,3),activation='relu'))

mymodel.add(MaxPooling2D())

mymodel.add(Flatten())

mymodel.add(Dense(1,activation='sigmoid'))

mymodel.compile(optimizer='Adam',loss='binary\_crossentropy',metrics=['accuracy'])

#Organize the data

train=ImageDataGenerator(rescale=1./255)

test=ImageDataGenerator(rescale=1./255)

train\_set=train.flow\_from\_directory('train',target\_size=(150,150),batch\_size=16,class\_mode='binary')

test\_set=test.flow\_from\_directory('test',target\_size=(150,150),batch\_size=16,class\_mode='binary')

#Train the model

k=mymodel.fit(train\_set,epochs=10,validation\_data=test\_set)

#Save the model

mymodel.save('drowsy.h5',k)

Main.py:

import streamlit as st

import cv2

from keras.models import load\_model

import numpy as np

from pygame import mixer

st.set\_page\_config(page\_title="Drowsiness Detection", page\_icon="drowsy.png")

st.title("DROWSINESS DETECTION SYSTEM")

st.sidebar.image("pageconfig.png")

mixer.init()

sound = mixer.Sound('alarm.mp3')

model = load\_model('drowsy.h5')

face\_cascade = cv2.CascadeClassifier('haar cascade files\haarcascade\_frontalface\_alt.xml')

lefteye\_cascade = cv2.CascadeClassifier('haar cascade files\haarcascade\_lefteye\_2splits.xml')

righteye\_cascade = cv2.CascadeClassifier('haar cascade files\haarcascade\_righteye\_2splits.xml')

font = cv2.FONT\_HERSHEY\_COMPLEX\_SMALL

score=0

eye\_status=['Closed','Open']

menu\_choice = st.sidebar.selectbox("MENU", ("HOME", "IP Camera", "WEB CAM"))

if menu\_choice == "HOME":

st.image("sleepy.gif")

st.write("Drowsiness Detection System is a Computer Vision Machine Learning Application which can be accessed through IP Camera

and can detect whether the person is drowsy or not.")

elif menu\_choice in ["WEB CAM", "IP Camera"]:

if menu\_choice == "WEB CAM":

camera\_option = st.text\_input("Enter 0 for Primary Camera or 1 for Secondary Camera")

else:

camera\_option = st.text\_input("Enter URL for the video")

btn = st.button("Start Camera")

if btn:

window = st.empty()

try:

video = cv2.VideoCapture(int(camera\_option)) if menu\_choice == "WEB CAM" else cv2.VideoCapture(camera\_option)

except:

st.write("Invalid camera input!")

st.stop()

btn2 = st.button("Stop Camera")

if btn2:

video.release()

st.experimental\_rerun()

while video.isOpened():

flag, frame = video.read()

if flag:

height, width = frame.shape[:2]

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(gray, minNeighbors=5, scaleFactor=1.1, minSize=(25, 25))

left\_eye = lefteye\_cascade.detectMultiScale(gray)

right\_eye = righteye\_cascade.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:

right\_eye = frame[y:y+h, x:x+w]

right\_eye = cv2.cvtColor(right\_eye, cv2.COLOR\_BGR2GRAY)

right\_eye = cv2.resize(right\_eye, (24, 24))

right\_eye = right\_eye/255

right\_eye = right\_eye.reshape(24, 24, -1)

right\_eye = np.expand\_dims(right\_eye, axis=0)

rmodel = model.predict(right\_eye)

rpred = np.argmax(rmodel, axis=1)

if rpred[0] == 1:

eye\_status = 'Open'

if rpred[0] == 0:

eye\_status = 'Closed'

break

for (x, y, w, h) in left\_eye:

left\_eye = frame[y:y+h, x:x+w]

left\_eye = cv2.cvtColor(left\_eye, cv2.COLOR\_BGR2GRAY)

left\_eye = cv2.resize(left\_eye, (24, 24))

left\_eye = left\_eye/255

left\_eye = left\_eye.reshape(24, 24, -1)

left\_eye = np.expand\_dims(left\_eye, axis=0)

lmodel = model.predict(left\_eye)

lpred = np.argmax(lmodel, axis=1)

if lpred[0] == 1:

eye\_status = 'Open'

if lpred[0] == 0:

eye\_status = 'Closed'

break

if rpred[0] == 0 and lpred[0] == 0:

score = score + 1

cv2.putText(frame, "Eyes are Closed",(10, height-20), font, 1, (255, 255, 255), 1, cv2.LINE\_AA)

else:

score = score - 5

cv2.putText(frame, "Eye is Open", (25, height-20), font, 1, (255, 255, 255), 1, cv2.LINE\_AA)

if score < 0:

score = 0

if score > 5:

try:

sound.play()

except:

pass

cv2.rectangle(frame, (0, 0), (width, height), (0, 0, 255), 4)

window.image(frame, channels='BGR')

cv2.destroyAllWindows()









