Janardhan Bhagat Shikshan Prasarak Sanstha's

CHANGU KANA THAKURARTS, COMMERCE, SCIENCE COLLEGE

NEW PANVEL(Autonomous)

PROJECT REPORT ON

"Driver Fatigue Detection"

DEVELOPED BY

Ms. Mohammad Naeem Raha

UNDER THE GUIDANCE OF

PROF.MR. Aniket

2021-2022

ACKNOWLEDGMENT

It gives me great pleasure to present my project on **“Driver Fatigue Detection”.** This is my first milestone in B.Sc Computer Science. It is always a difficult task to acknowledge all those who helped me a lot doing this project. Never the less I have made an attempt through this report to express my deepest gratitude to all those who have contributed in making of this project either directly or indirectly.

I would extend my thanks to our principal **Dr.V.D.Barahate.** for his support and facilities given to us for the same.

I am also thankful to my department head **Prof. Mrs.P.M.Jadhav** who encourage me and gave me moral support during my project. I would like to thank all my teaching and non-teaching staff of computer science faculty who helped me in completion of my project.

I would like to express my sincere thanks to my project guide **prof. Mr. Aniket** who helped me throughout the project.

INDEX

INTRODUCTION

The number of cars is increasing exponentially worldwide and everyday there are more and more people purchasing new cars and adding to the automotive population. However, with the increased number of cars the risk of accidents is also increasing.

According WHO every year, 1.3 million people die as a result of road traffic crashes and it costs every respective country 3% of their gross domestic products. Besides, WHO reports has classified drivers' carelessness, sleeplessness and alcoholism as one of the main reason of crashes on the roads.

Road safety has always been a concern for most countries which has lead to development of intelligent transportation system such as cruise control, park assistance control, pedestrian detection system, blind spot detection system and many more technologies.

Considering that driver carelessness is one of the main contributors to crashes on the road, this project is aiming to minimize its risk and it should be inbuilt with the vehicle. A driver fatigue detection system's aim is to monitor the driver while driving and raise an alarm if the driver was not paying fully attention to the road or was drowsy.

LITRARURE REVIEW

In road safety and driving vehicle, driver monitoring is an important topic. Many projects have been developed for driver monitoring and detection using different technologies. Some possible techniques for monitoring are sensing of physiological characteristics, driver operation, vehicle response and driver response. Though techniques are more accurate but it is not realistic since it requires sensing electrodes to be connected to the driver which itself can be annoying and distracting.

The technique which relies on monitoring the eyes of driver is well suited to monitor the driver. In this technique the system detects and monitors the eyes of the driver and its closure to decide whether to raise an alarm or not.

The research on this technique has been started years ago, however very few commercial systems has been released. The driver drowsiness detection of Volvo and Mercedes Benz are two of these systems, though they can only be seen in high end vehicles.

PROPOSED SYSTEM

Several different algorithms can be used for eye tracking and monitoring. Some of these technologies relate to features of the eye (reflection of eye) of a video image of the driver. However, deep learning is used in this project to develop an intelligent model based on thousands of people faces, for constantly tracking drivers' eyes and deciding if the driver's eyes is closed or open.

Applying this model on live video which is captured from the driver will help with calculation of eye closure time. Considering that eye closure time of a drowsy driver is more than the normal blinking time which can lead disastrous accidents. Therefore, we will warn the driver as soon as closed eye is detected.

WORKING OF SYSTEM

SENSING PHASE:

A camera is used to take live video of the driver and save it as different frames.

EYE DETECTION:

The Har Cascade eye classifier is used to detect the eye.

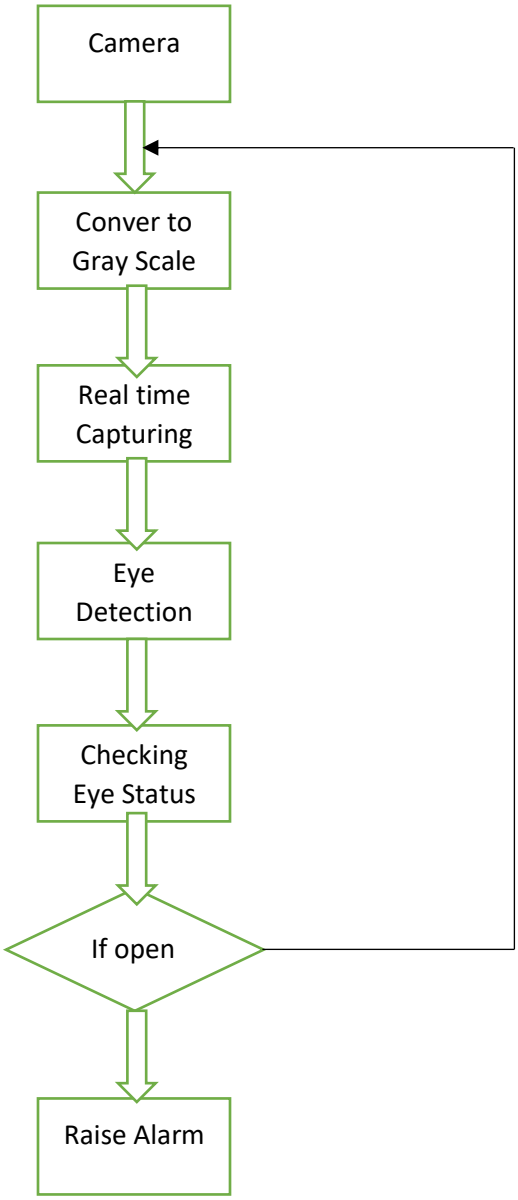
EYE STATUS:

The model is used on eye detector to predict whether eyes closed or open and store its value.

RAISE WARNING:

Using the stored value, the system decides if the driver is drowsy or not and it will raise an alarm if driver is classified as drowsy.

PROPOSED SYSTEM'S DIAGRAM:



SYSTEM REQUIRMENT

Hardware Requirements:

- Camera Module
- A computer such as (Raspberry Pi3)
- Speaker

Software Requirements:

- Jupyter notebook
- Python 3.6
- Libraries (TensorFlow, open cv, Keras, OS, Pygame, Time)
- Operating system

SYSTEM IMPLEMETATION

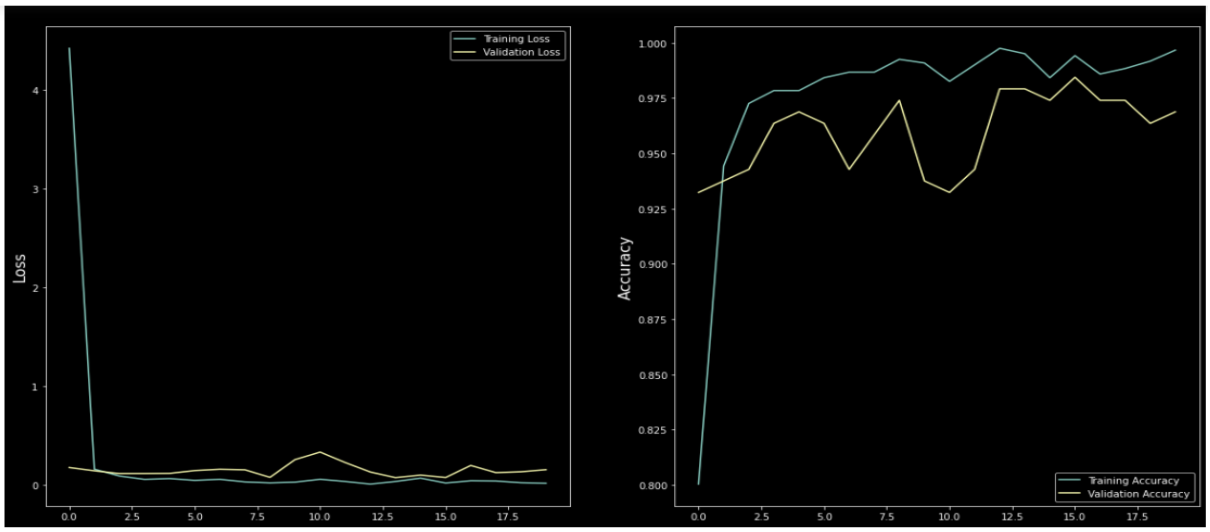
modelTraining.ipynb

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import os

# Importing Deep Learning Libraries
from tensorflow.keras.optimizers import Adam,SGD,RMSprop
from keras.preprocessing.image import load_img, img_to_array
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import
Dense,Input,Dropout,GlobalAveragePooling2D,Flatten,Conv2D,BatchNormalization,Activ
ation,MaxPooling2D
from keras.models import Model,Sequential
from keras.preprocessing import image


def generator(dir, gen= image.ImageDataGenerator(rescale=1./255),
shuffle=True,batch_size=1,target_size=(24,24),class_mode='categorical' ):

   return
gen.flow_from_directory(dir,batch_size=batch_size,shuffle=shuffle,color_mode='grayscal
e',class_mode=class_mode,target_size=target_size)


batch_size  = 32
picture_size = 24
datagen_train  = ImageDataGenerator()
datagen_val = ImageDataGenerator()

train_set = datagen_train.flow_from_directory("dataset_new/train",
                                              target_size = (picture_size,picture_size),
                                              color_mode = "grayscale",
                                              batch_size=batch_size,
                                              class_mode='categorical',
                                              shuffle=True)
```

```
test_set = datagen_val.flow_from_directory(f"dataset_new/test",
                                           target_size = (picture_size,picture_size),
                                           color_mode = "grayscale",
                                           batch_size=batch_size,
                                           class_mode='categorical',
                                           shuffle=True)
SPE= len(train_set.classes)//batch_size #Steps_per_Epochs
VS = len(test_set.classes)//batch_size #Validation_size
print(SPE,VS)

model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(24,24,1)),
    MaxPooling2D(pool_size=(1,1)),
    Conv2D(32,(3,3),activation='relu'),
    MaxPooling2D(pool_size=(1,1)),
#32 convolution filters used each of size 3x3
#again
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(1,1)),

#64 convolution filters used each of size 3x3
#choose the best features via pooling

#randomly turn neurons on and off to improve convergence
    Dropout(0.25),
#flatten since too many dimensions, we only want a classification output
    Flatten(),
#fully connected to get all relevant data
    Dense(128, activation='relu'),
#one more dropout for convergence' sake :)
    Dropout(0.5),
#output a softmax to squash the matrix into output probabilities
    Dense(2, activation='softmax')
])
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
model.summary()

history = model.fit_generator (generator=train_set,
                               steps_per_epoch=SPE,
```

```
                    epochs=20,
                    validation_data = test_set,
                    validation_steps = VS,
                    )

model.save('models/best1.h5')


plt.style.use('dark_background')

plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : Adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
```

Model Accuracy and loss



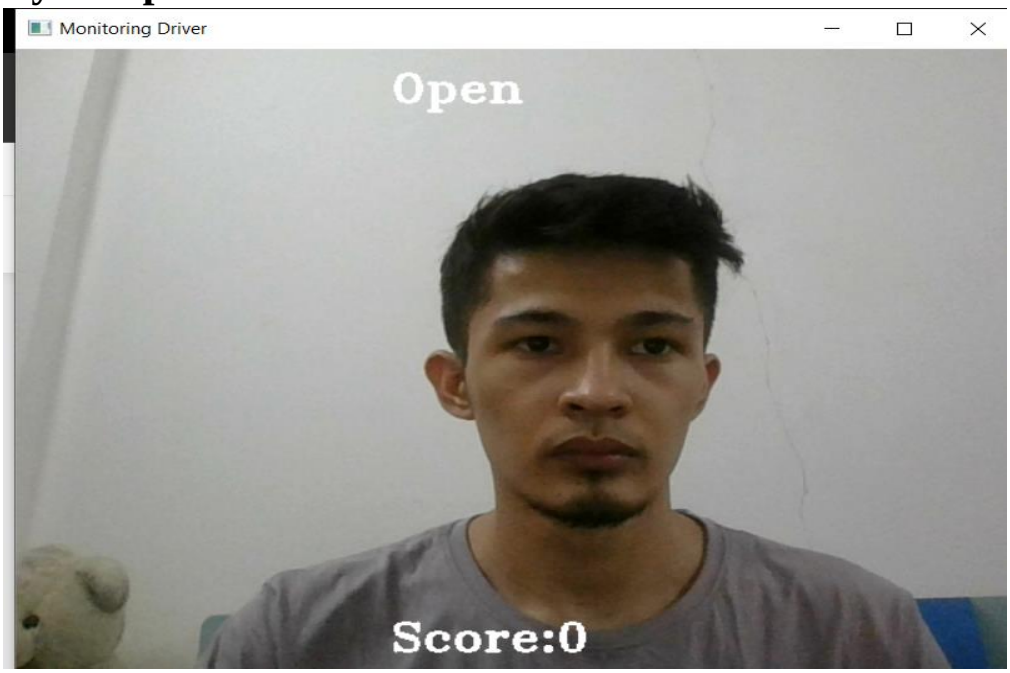
drow.dtect.ipynb

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


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


leye = cv2.CascadeClassifier('haar/haarcascade_lefteye_2splits.xml')
reye = cv2.CascadeClassifier('haar/haarcascade_righteye_2splits.xml')
rpred=[99]
lpred=[99]

lbl=['Closed','Open']

model = load_model('models/best1.h5')
cap = cv2.VideoCapture(0)
font = cv2.FONT_HERSHEY_TRIPLEX
count=0
score=0


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


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


    left_eye = leye.detectMultiScale(gray) #Left eye detection
    right_eye =  reye.detectMultiScale(gray) #Right eye detection


    for (x,y,w,h) in right_eye:
        rightEye=frame[y:y+h,x:x+w]
        count=count+1
        rightEeye = cv2.cvtColor(rightEye,cv2.COLOR_BGR2GRAY)
```

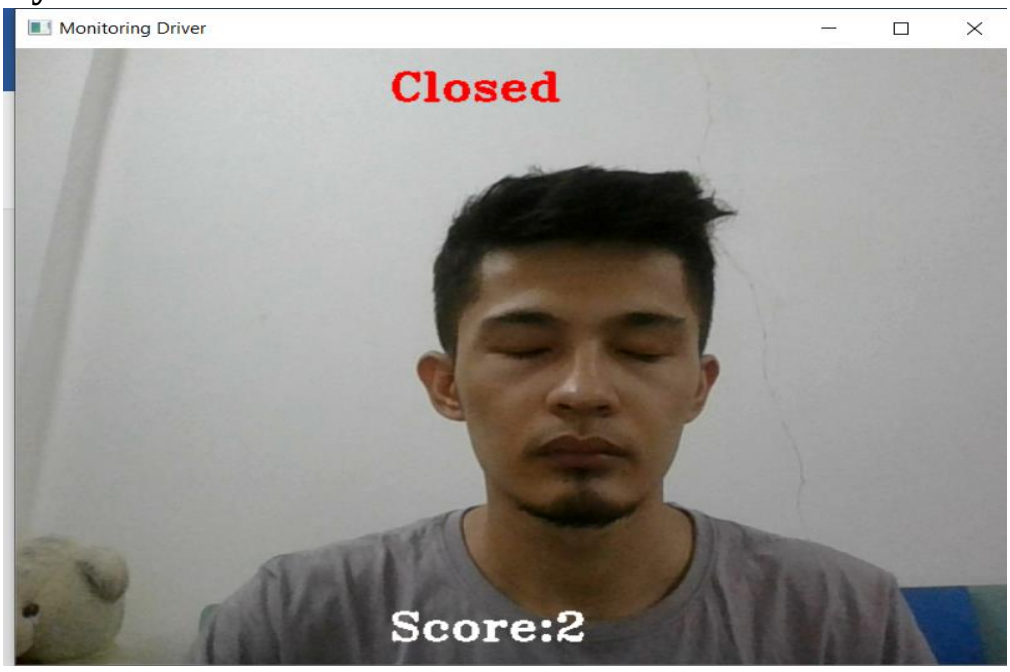
```
        rightEye = cv2.resize(rightEye,(24,24))
        rightEye= rightEye/255
        rightEye=  rightEye.reshape(24,24,-1)
        rightEye = np.expand_dims(rightEye,axis=0)
        rpred = np.argmax(model.predict(rightEye), axis=-1)
        if(rpred[0]==1):
            lbl='Open'
        if(rpred[0]==0):
            lbl='Closed'
        break

    for (x,y,w,h) in left_eye:
        leftEye=frame[y:y+h,x:x+w]
        count=count+1
        leftEye = cv2.cvtColor(leftEye,cv2.COLOR_BGR2GRAY)
        leftEye = cv2.resize(leftEye,(24,24))
        leftEye= leftEye/255
        leftEye=leftEye.reshape(24,24,-1)
        leftEye = np.expand_dims(leftEye,axis=0)
        lpred = np.argmax(model.predict(leftEye), 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
        cv2.putText(frame,"Closed",(240,40), font, 1,(0,0,255),2)
    else:
        score=score-1
        cv2.putText(frame,"Open",(240,40), font, 1,(255,255,255),2)

    if(score<0):
        score=0
    cv2.putText(frame,'Score:'+str(score),(240,height-20), font, 1,(255,255,255),2)
    if(score>15):
        sound.play()
    cv2.imshow('Monitoring Driver',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

RESULT:

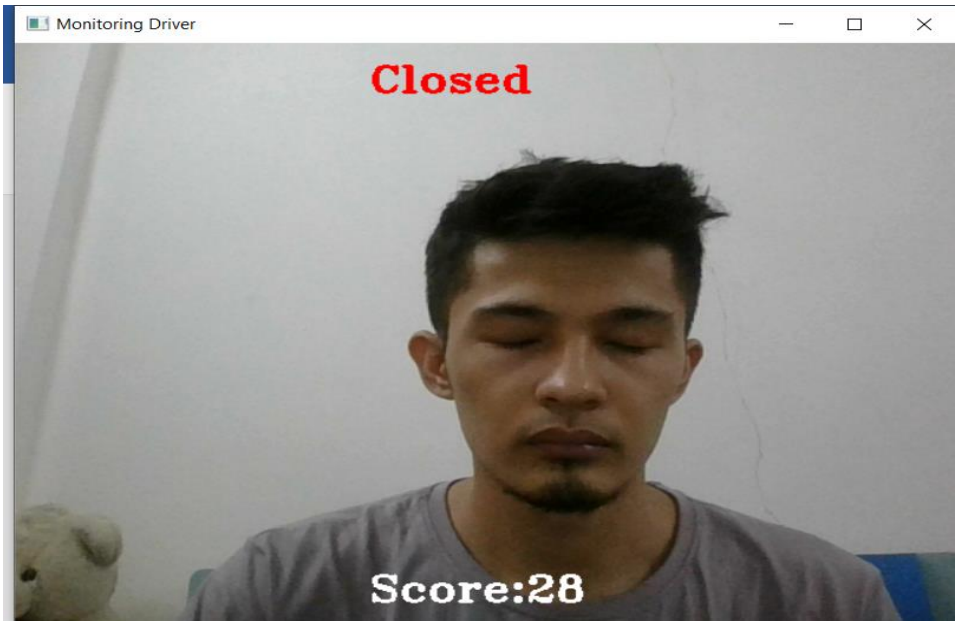
Eye is open



Eye is closed



Eye is closed and the score has passed the limit (warning the driver)



CONCLUSION

The driver fatigue detestation system is detecting drowsiness in driver accurately and instantly and raise a warning when needed. Driver fatigue system is developed based on eye closure time period which can be used to differentiate between blinking and drowsiness. The system can be useful to prevent accidents due to sleeplessness of the driver which works well even if the driver is wearing specs or the lighting condition is not good.

Information about the eye position is obtained by use of Har Cascade eye detection model. While monitoring the driver the developed model is applied on the derived information to predict whether eye is open or closed. The system will take the information and raise a warning if the eye was closed for too long.

REFERENCE

- https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries
- https://data-flair.training/blogs/python-project-driver-drowsiness-detection-system/
- https://www.tensorflow.org/js/guide/train_models
- https://www.kaggle.com/serenaraju/yawn-eye-dataset-new