VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF ELECTRONICS ENGINEERING

Department of Embedded Technology

Driver Drowsiness Detection

Name
Darsh Gupta
20BEC0563

Project Report
of
ECE 4032 – Neural Networks & Deep Learning

Fall Sem 2020-21

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF ELECTRONICS ENGINEERING

Department of SENSE

Driver Drowsiness Detection

Darsh Gupta
(20BEC0563)

Project Report
of
ECE 4032 – Neural Networks & Deep Learning

Fall Sem 2020-21

Submitted to
Faculty: Dr. R. Sujatha
Date: 9th December, 2021
Slot: G1+TG1

Index

Abstract

Driver drowsiness has become one of the main reasons for large number of road accidents. The main aim of this project is to overcome the problem of road accidents which are related to drivers experiencing fatigue leads to a need arises to design a system that keeps the driver focused on the road. With the evolution and improvement in Computer Vision technologies, smart/intelligent cameras are developed to predict drowsiness in drivers, thereby alerting drivers which in turn reduce accidents when they are feeling drowsy. In this work, a new approach is taken using deep learning to detect driver drowsiness based on Eye state while driving the vehicle. To detect the face and extract the eye region from the face images, OpenCV algorithms and haar cascade classifier are used.

Introduction

In a car safety technology, driver drowsiness detection is very essential to prevent road accidents. Now-a-days, many people using automobiles for daily commutation, higher living standards, comfortability, and timing constraints to reach destinations. According to the National Highway Traffic Safety Administration, every year about 1,00,000 police-reported crashes involves drowsy driving. These crashes result in more than 1,550 fatalities and 71,000 injuries. Sleep-deprived drivers remain responsible for about 40% of the road accidents, according to enforcement officers patrolling the highways and major roads here.

Exhausted drivers who doze off at the wheel are responsible for about 40% of road accidents, says a study by the Central Road Research Institute (CRRI).

There are different signs of driver drowsiness can be observed while driving the vehicle such as inability to keep eyes open, frequently yawning, moving the head forward etc.

To determine the level of driver drowsiness various measures are used. These measures are Physiological Measures, Behavioral Measures and Vehicle-based Measures.

In physiological measures, Electrocardiography (ECG), Electroencephalography (EEG), and Electrooculogram (EOG) are used to access the drivers conditions. Even though these devices provide accurate results, due to their practical limitations, these are not widely accepted. In vehicle based measures, drowsiness is analyzed based on steering wheel movements and braking patterns. Face detection Algorithms was used to identify face regions from the input images in Face Detection phase

Literature Review

1) A Realistic Dataset and Baseline Temporal Model for Early Drowsiness Detection (15th April 2019)

In this paper, we presented a new and publicly available real-life drowsiness dataset (RLDD), which, to the best of our knowledge, is significantly larger than existing datasets, with almost 30 hours of video. We have also proposed an end-to-end baseline method using the temporal relationship between blinks for multistage drowsiness detection. The proposed method has low computational and storage demands. Our results demonstrated that our method outperforms human judgment in two designed metrics on the RLDD dataset

2) Drowsiness Detection with Machine Learning (14th December 2019)

First, simpler models can be just as efficient at completing tasks as more complex models. In our case, the K-Nearest Neighbor model gave an accuracy similar to the LSTM model. However, because we do not want to misclassify people who are drowsy as alert, ultimately it is better to use the more complex model with a lower false-negative rate than a simpler model that may be cheaper to deploy. Second, normalization was crucial to our performance. We recognized that everybody has a different baseline for eye and mouth aspect ratios and normalizing for each participant was necessary. Outside of runtime for our models, data pre-processing and feature extraction/normalization took up a bulk of our time. It will be interesting to update our project and look into how we can decrease the false-negative rate for kNN and other simpler models.

3) Vision-based drowsiness detector for a Realistic Driving Simulator (20th September,2010)

This paper presents a non intrusive approach for monitoring driver drowsiness, based on computer vision techniques, installed in a realistic driving simulator. The proposed drowsiness detection method has demonstrated to be valid, showing an 85% of awake state recall rate considering only PERCLOS. The computer vision and the Gaussian model method does not need fixed threshold to determine the eye opening. However the PERCLOS signal assessment needs a nominal or fixed eye opening threshold value. This nominal value is dependent on the constraints of the driver's eye and it is calculated in an initial automatic process

4) Driver Fatigue Detection Based on Convolutional Neural Networks Using EM-CNN (18th November 2020)

With a focus on fatigue driving detection research, a fully automated driver fatigue status detection algorithm using driving images is proposed. In the proposed algorithm, the multitask cascaded convolutional network (MTCNN) architecture is employed in face detection and feature point location, and the region of interest (ROI) is extracted using feature points. A convolutional neural network, named EM-CNN, is proposed to detect the states of the eyes and mouth from the ROI images. The percentage of eyelid closure over the pupil over time (PERCLOS) and mouth opening degree (POM) are two parameters used for fatigue detection. Experimental results demonstrate that the proposed EM-CNN can efficiently detect driver fatigue status using driving images. The proposed algorithm EM-CNN outperforms other CNN-based methods, i.e., AlexNet, VGG-16, GoogLeNet, and ResNet50, showing accuracy and sensitivity rates of 93.623% and 93.643%, respectively.

5) A Machine-Learning Approach for Driver-Drowsiness Detection based on Eye-State (13th April, 2021)
The drowsiness detection based on eye state has been done accurately based on the varying features and factors, and also with the help of experts knowledge. Predicting the facial landmarks and detecting the eye-state and displaying the driver status on the screen and in the App is the most necessity ingredient for drowsiness detection. Generally, the driving person feels drowsy due to continues driving for long hours or Physical illness or might be drunken and this leads to major road accidents. Our aim is to detect the drowsiness, make them alert to prevent accidents and generate a notification in the app and an alarm sound.

6) Driver drowsiness detection using ANN image processing (October, 2017)
Analyzing the results of applying these networks on the acquisitioned images it can be concluded that both networks had very good results, with 100% positive classification results, which were presented in chapter 3. The small number of neurons used in the hidden layers to successfully classify the images (10 for the 1 hidden layer network and 15 for the autoencoder network) allows the implementation of these networks on compact computing devices, using a very small portion of their memory. Also the processing time is in the order of milliseconds on a Windows based computer, which on a compact device can be more reduced. The training of the network can be done specifically for each driver, thus enhancing the classification success rate

7) Robust Drowsiness Detection for Vehicle Driver using Deep Convolutional Neural Network (3rd June, 2020)
During facial features extractions proposed method used aggregation strategy and integral image construction to process rectangle features in case face region cannot be extracted due to some issues, i.e. light reflection, shadow in input frames. In face alignment phase, proposed method used cascade of regressors cutting edge method in order to improve identification of facial landmarks under highly varying lighting conditions for video frames. Later, in pupil detection step, proposed research used deep convolutional neural network (DCNN) for accurate pupil detection for nonlinear data pattern where proposed method used facial landmarks as the center of segmentation to be used through DCNN network.

8) Detecting Human Driver Inattentive and Aggressive Driving Behavior Using Deep Learning: Recent Advances, Requirements and Open Challenges (3rd June, 2020)
In this study, we classified and discussed the human driver's Inattentive driving behavior (HIDB) into two major categories; Distraction and Fatigue/Drowsiness. The detection of driver distraction and driver fatigue/drowsiness was classified according to the features selected for the detection of human driving behavior in the literature. Aggressive driving behavior being another more risky human driving behavior was also explained and discussed, high-lighting the causes and effects of different aggressive driving styles on human safe driving

9) Detecting Drowsy Drivers Using Machine Learning Algorithms (Decemeber, 2020)
In this work, we tried to detect drowsy drivers using supervised machine learning algorithms. Because of the time-series nature of the data, we had to do aggregation over the time-series to generate features. We tried to generate aggregate features in the granularity of per-run and per-event in each run. We found that the per-event aggregate features result to better classifiers with respect to area under ROC curve. The per-event aggregate features are also more informative and

intuitive when analyzing the selected attributes from the dataset. Also, function-based classifiers such as Logistic Regression and SMO performed better than Bayes and Trees for this classification task.

10) An efficient detection approach of driver drowsiness using multiple convolution haar cascade kernelized CNN(MCHCKCNN) Algorithm (March, 2021)

In this field, we are analyzed different experimentation for Real-Time DDD System. In a practical simulation setting, we took careful care to carry out our test testing. For a practical mode of vehicular mobility. With python as a simulation tool, we analyzed our model. While certain traditional models can detect the positions of several facials, the eyes and mouth areas of the driver cannot be established. However, the driver will practically have diverse and complex facial expressions that distort their detection.

11) Real-time Driver Drowsiness Detection for Android Application Using Deep Neural Networks Techniques (2018)

This paper proposes a drowsiness detection system based on multilayers perceptron classifiers. It is specifically designed for embedded systems such as Android mobile. The role of the system is to detect facial landmark from images and deliver the obtained data to the trained model to identify the driver's state. The purpose of the method is to reduce the model's size considering that current applications cannot be used in embedded systems due to their limited calculation and storage capacity. According to the experimental results, the size of the used model is small while having the accuracy rate of 81%. Hence, it can be integrated into advanced driver-assistance systems, the Driver drowsiness detection system, and mobile applications. However, there is still space for the performance improvement. The further work will focus on detecting the distraction and yawning of the driver.

Hardware and Software Details

1) Web Camera
For capturing images/video

2) Python
Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems.

3) OpenCV
OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

4) TensorFlow
It is an open-source artificial intelligence library, using data flow graphs to build models. It allows developers to create large-scale neural networks with many layers. TensorFlow is mainly used for: Classification, Perception, Understanding, Discovering, Prediction and Creation.

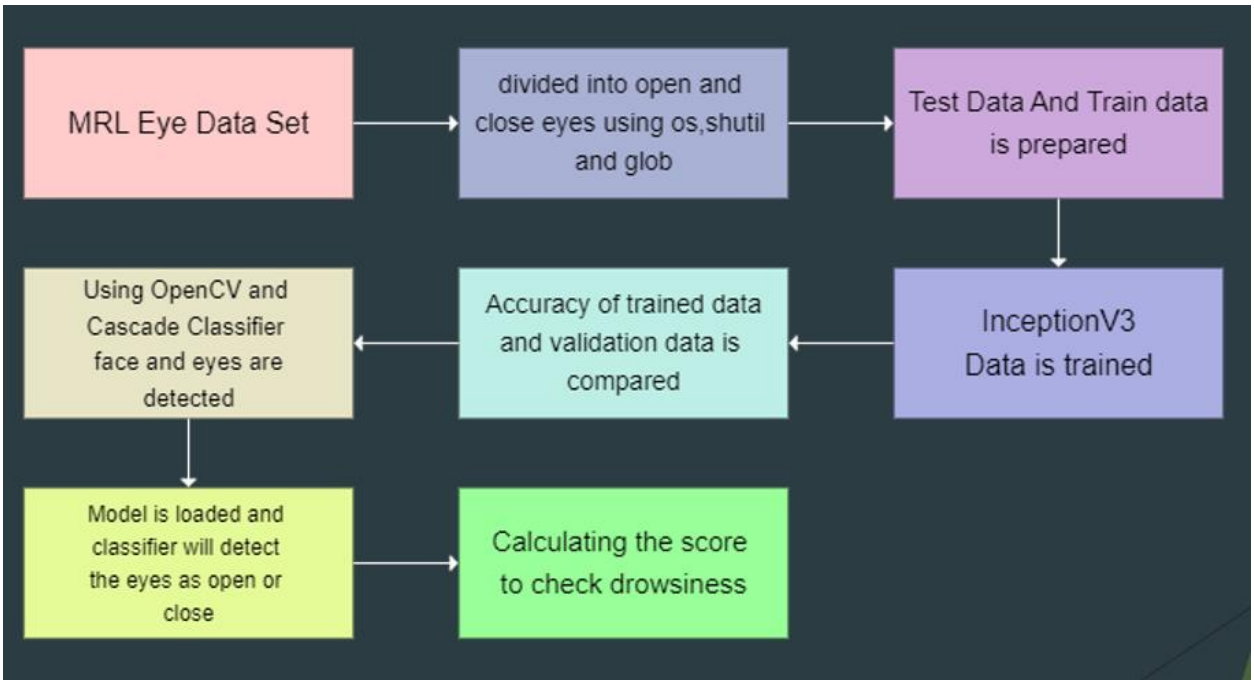
5) Keras
Keras is one of the leading high-level neural networks APIs. It is written in Python and supports multiple back-end neural network computation engines.

6) Pygame
Python is the most popular programming language or nothing wrong to say that it is the next-generation programming language. In every emerging field in computer science, Python makes its presence actively. Python has vast libraries for various fields such as Machine Learning (Numpy, Pandas, Matplotlib), Artificial intelligence (Pytorch, TensorFlow), and Game development (Pygame,Pyglet)**.**

7) InceptionV3
CNN Architecture trained on ImageNet Dataset. Inception-V3 is a convolution neural network that is 48 layers deep. You can load a pretrained version of the network on more than a million images from the ImageNet Dataset. The pretrained network can classify images into 1000 object categories, such as keywords, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 299-by-299

8) MRL eye dataset
MRL Eye Dataset, the large-scale dataset of human eye images. This dataset contains infrared images in low and high resolution, all captured in various lightning conditions and by different devices. The dataset is suitable for testing several features or trainable classifiers. In order to simplify the comparison of algorithms, the images are divided into several categories, which also makes them suitable for training and testing classifiers.

Block Diagram



Problem Statement

According to the Centers for Disease Control and Prevention, about 1 in 25 adult drivers report having fall asleep while driving in the previous 30 days, and many more admit to driving when they were sleep-deprived.

These startling figures show how prevalent drowsy driving is. What drivers may not realize is how much drowsy driving puts themselves – and others – at risk. In fact, an estimated 64,00 people died annually in crashes involving drowsy driving, according to the National Sleep Foundation.

The National Highway Traffic Safety Administration estimates that every year about 10,000 police-reported, drowsy-driving crashes result in nearly 800 fatalities and about 50,000 injuries. The real number may be much higher, however, as it is difficult to determine whether a driver was drowsy at the time of a crash.

A study by the AAA Foundation for Traffic Safety estimated that 3,28,000 drowsy driving crashes occurs annually That's more than three times the police-reported number. The same study found that 109,000 of those drowsy driving crashes resulted in an injury and about 6,400 were fatal. The researchers suggest the prevalence of drowsy driving fatalities is more than 350% greater than reported.

Beyond the human toll is the economic one. NHTSA estimates fatigue-related crashes resulting in injury or death cost society $109 billion annually, not including property damage.

Clearly, these drowsy drivers can cause fatal accidents on roads.

Proposed Work to overcome the Problem Mentioned

The drowsiness detection based on eye state has been done accurately based on the varying features and factors, and also with the help of experts knowledge. Predicting the facial landmarks and detecting the eye-state and displaying the driver status on the screen and producing alert sound if the driver is drowsy for drowsiness detection.

Generally, the driving person feels drowsy due to continues driving for long hours or Physical illness or might be drunken and this leads to major road accidents. Our aim is to detect the drowsiness, make them alert to prevent accidents and generate an alarm sound.

Project Description

Data preparation

MRL eye dataset is used in this project. It is large-scale dataset of human eye images. This dataset contains infrared images in low and high resolution, all captured in various lightning conditions and by different devices. The dataset is suitable for testing several features or trainable classifiers. In order to simplify the comparison of algorithms, the images are divided into several categories, which also makes them suitable for training and testing classifiers.
In the dataset, we annotated the following properties (the properties are indicated in the following order):
subject ID; in the dataset, we collected the data of 37 different persons (33 men and 4 women)
image ID; the dataset consists of 84,898 images
gender [0 - man, 1 - woman]; the dataset contains the information about gender for each image (man, woman)
glasses [0 - no, 1 - yes]; the information if the eye image contains glasses is also provided for each image (with and without the glasses)
eye state [0 - closed, 1 - open]; this property contains the information about two eye states (open, close)
reflections [0 - none, 1 - small, 2 - big]; we annotated three reflection states based on the size of reflections (none, small, and big reflections)
lighting conditions [0 - bad, 1 - good]; each image has two states (bad, good) based on the amount of light during capturing the videos
sensor ID [01 - RealSense, 02 - IDS, 03 - Aptina]; at this moment, the dataset contains the images captured by three different sensors (Intel RealSense RS 300 sensor with 640 x 480 resolution, IDS Imaging sensor with 1280 x 1024 resolution, and Aptina sensor with 752 x 480 resolution)

The **OS module** in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality.

The **shutil** in Python is a module that offers several functions to deal with operations on files and their collections. It provides the ability to copy and removal of files.

Glob is used to find the files and folders whose names follow a specific pattern. The searching rules are similar to the Unix Shell path expansion rules.

tqdm is a Python library that allows you to output a smart progress bar by wrapping around any iterable. A **tqdm** progress bar not only shows you how much time has elapsed, but also shows the estimated time remaining for the iterable.

Open eyes and closed eye data was separated and moved 90 percent of the images was moved to train dataset and remaining to test dataset manually

Model Training

InceptionV3 CNN Architecture Trained on ImageNet Data Set. Inception-v3 is a convolutional neural network that is 48 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 299-by-299.

Transfer learning, used in machine learning, is the reuse of a pre-trained model on a new problem. In transfer learning, a machine exploits the knowledge gained from a previous task to improve generalization about another. For example, in training a classifier to predict whether an image contains food, you could use the knowledge it gained during training to recognize drinks.

CNN – Image data pre-processing with generators

- Read the picture files (stored in data folder).
- Decode the JPEG content to RGB grids of pixels with channels.
- Convert these into floating-point tensors for input to neural nets.
- Rescale the pixel values (between 0 and 255) to the [0, 1] interval (as training neural networks with this range gets efficient).

Keras has a module with image-processing helping tools, located at keras.preprocessing.image. It contains the class *ImageDataGenerator*, which lets you quickly set up Python generators that can automatically turn image files on disk into batches of preprocessed tensors.

Image augmentation is a technique of applying different transformations to original images which results in multiple transformed copies of the same image. Each copy, however, is different from the other in certain aspects depending on the augmentation techniques you apply like shifting, rotating, flipping, etc.

Applying these small amounts of variations on the original image does not change its target class but only provides a new perspective of capturing the object in real life. And so, we use it is quite often for building deep learning models.

Batch size (8): The batch size is the number of samples processed before updating the model. The number of epochs represents the total number of passes through the training dataset.

Epoch (5): It indicates the number of passes of the entire training dataset the machine learning algorithm has completed

Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments.

Accuracy achieved – 92.03 percent

Main Program

OpenCV By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

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.

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 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**.

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.

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.
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()**

Screenshots of the prototype

Data preparation

In [12]:
```
import os
import shutil
import glob
from tqdm import tqdm
```

subject ID: xxx

image number: xxx

gender: 0 - male 1 - famale

glasses: 0 - no 1 - yes

eye state: 0 - close 1 - open

reflections: 0 - none 1 - low 2 - high

lighting conditions/image quality: 0 - bad 1 - good

sensor type: 01 - RealSense SR300 640x480 02 - IDS Imaging, 1280x1024 03 - Aptina Imagin 752x480

example: s001_00123_0_0_0_0_0_01.png

In [3]:
```
Raw_DIR= r'D:\Driver Drowsiness detection\mrlEyes_2018_01\mrlEyes_2018_01'
for dirpath, dirname, filenames in os.walk(Raw_DIR):
    for i in tqdm([f for f in filenames if f.endswith('.png')]):
        if i.split('_')[4]=='0':
            shutil.copy(src=dirpath+'/'+i, dst=r'D:\Driver Drowsiness detection\Preparad_Data\Close Eyes')

        elif i.split('_')[4]=='1':
            shutil.copy(src=dirpath+'/'+i, dst=r'D:\Driver Drowsiness detection\Preparad_Data\Open Eyes')
```

```
0it [00:00, ?it/s]
100%|██████████| 3242/3242 [00:29<00:00, 110.13it/s]
100%|██████████| 1114/1114 [00:09<00:00, 117.14it/s]
100%|██████████| 679/679 [00:06<00:00, 111.79it/s]
100%|██████████| 1069/1069 [00:09<00:00, 115.52it/s]
```

Model Training

In [4]:
```
import tensorflow as tf
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dropout,Input,Flatten,Dense,MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator  # Data Augumentation
```

In [5]:
```
tf.test.is_gpu_available()
```

```
WARNING:tensorflow:From C:\Users\darsh\AppData\Local\Temp/ipykernel_2664/337460670.py:1: is_gpu_available (from tensorflow.python.framework.test_util) i
s deprecated and will be removed in a future version.
Instructions for updating:
Use `tf.config.list_physical_devices('GPU')` instead.
```

Out[5]:
```
False
```

In [6]:
```
batchsize=8
```

In [7]:
```
train_datagen= ImageDataGenerator(rescale=1./255, rotation_range=0.2,shear_range=0.2,
    zoom_range=0.2,width_shift_range=0.2,
    height_shift_range=0.2, validation_split=0.2)

train_data= train_datagen.flow_from_directory(r'D:\Driver Drowsiness detection\Preparad_Data\Train',
                                target_size=(80,80),batch_size=batchsize,class_mode='categorical',subset='training' )

validation_data= train_datagen.flow_from_directory(r'D:\Driver Drowsiness detection\Preparad_Data\Train',
                                target_size=(80,80),batch_size=batchsize,class_mode='categorical', subset='validation')
```

```
Found 63292 images belonging to 2 classes.
Found 15822 images belonging to 2 classes.
```

In [8]:
```
test_datagen = ImageDataGenerator(rescale=1./255)

test_data = test_datagen.flow_from_directory(r'D:\Driver Drowsiness detection\Preparad_Data\Test',
                                target_size=(80,80),batch_size=batchsize,class_mode='categorical')
```

```
Found 4752 images belonging to 2 classes.
```

In [9]:
```
bmodel = InceptionV3(include_top=False, weights='imagenet', input_tensor=Input(shape=(80,80,3)))
hmodel = bmodel.output
hmodel = Flatten()(hmodel)
hmodel = Dense(64, activation='relu')(hmodel)
hmodel = Dropout(0.5)(hmodel)
hmodel = Dense(2,activation= 'softmax')(hmodel)

model = Model(inputs=bmodel.input, outputs= hmodel)
for layer in bmodel.layers:
    layer.trainable = False
```

In [10]:
```
model.summary()
```

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 80, 80, 3)]  0           []

 conv2d (Conv2D)                (None, 39, 39, 32)   864         ['input_1[0][0]']

 batch_normalization (BatchNorm  (None, 39, 39, 32)  96          ['conv2d[0][0]']
 alization)

 activation (Activation)        (None, 39, 39, 32)   0           ['batch_normalization[0][0]']

 conv2d_1 (Conv2D)              (None, 37, 37, 32)   9216        ['activation[0][0]']

 batch_normalization_1 (BatchNo  (None, 37, 37, 32)  96          ['conv2d_1[0][0]']
 rmalization)

 activation_1 (Activation)      (None, 37, 37, 32)   0           ['batch_normalization_1[0][0]']

 conv2d_2 (Conv2D)              (None, 37, 37, 64)   18432       ['activation_1[0][0]']

 batch_normalization_2 (BatchNo  (None, 37, 37, 64)  192         ['conv2d_2[0][0]']
 rmalization)

 activation_2 (Activation)      (None, 37, 37, 64)   0           ['batch_normalization_2[0][0]']
```

Model Evaluation

In [17]:
```
acc_tr, loss_tr = model.evaluate_generator(train_data)
print(acc_tr)
print(loss_tr)
```

```
C:\Users\darsh\AppData\Local\Temp/ipykernel_2664/713691994.py:1: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a futu
ersion. Please use `Model.evaluate`, which supports generators.
  acc_tr, loss_tr = model.evaluate_generator(train_data)
0.13321372866630554
0.9485716819763184
```

In [18]:
```
acc_vr, loss_vr = model.evaluate_generator(validation_data)
print(acc_vr)
print(loss_vr)
```

```
C:\Users\darsh\AppData\Local\Temp/ipykernel_2664/4081756742.py:1: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a futu
version. Please use `Model.evaluate`, which supports generators.
  acc_vr, loss_vr = model.evaluate_generator(validation_data)
0.1836639791727066
0.9230185747146606
```

In [19]:
```
acc_test, loss_test = model.evaluate_generator(test_data)
print(acc_tr)
print(loss_tr)
```

Main code

```
import cv2
import tensorflow as tf
from tensorflow.keras.models import load_model
import numpy as np
from pygame import mixer
```

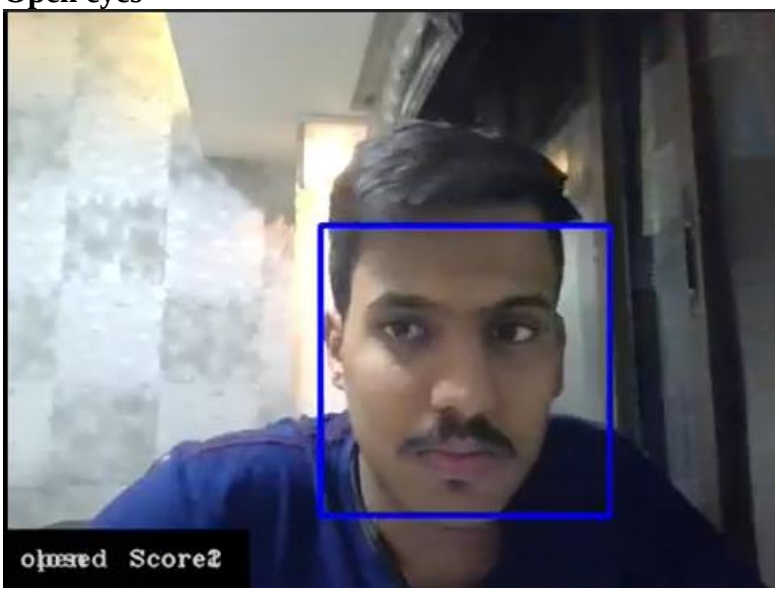
```
pygame 2.1.0 (SDL 2.0.16, Python 3.9.7)
Hello from the pygame community. https://www.pygame.org/contribute.html
```

```
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')
model = load_model(r'D:\Driver Drowsiness detection\models\model.h5')
```

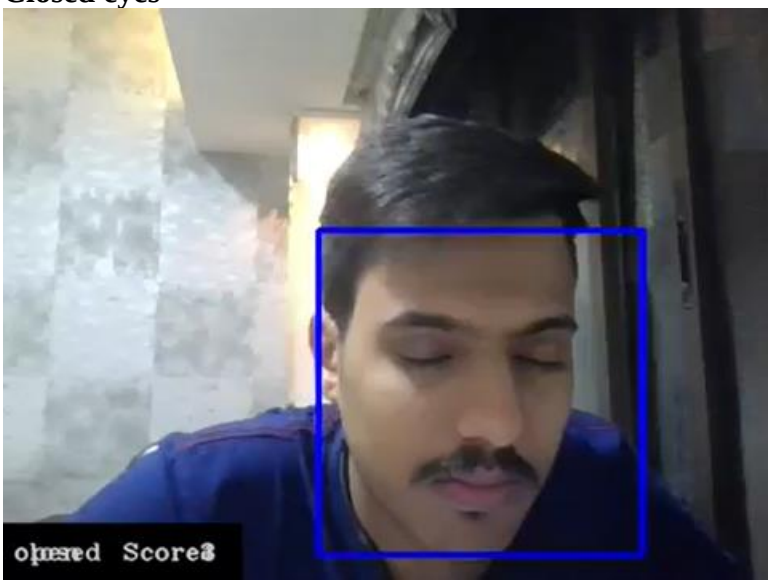
```
mixer.init()
sound= mixer.Sound(r'D:\Driver Drowsiness detection\alarm.wav')
cap = cv2.VideoCapture(0)
Score = 0
while True:
    ret, frame = cap.read()
    height,width = frame.shape[0:2]
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces= face_cascade.detectMultiScale(gray, scaleFactor= 1.2, minNeighbors=3)
    eyes= eye_cascade.detectMultiScale(gray, scaleFactor= 1.1, minNeighbors=1)

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

    for (x,y,w,h) in faces:
        cv2.rectangle(frame,pt1=(x,y),pt2=(x+w,y+h), color= (255,0,0), thickness=3 )

    for (ex,ey,ew,eh) in eyes:
        #cv2.rectangle(frame,pt1=(ex,ey),pt2=(ex+ew,ey+eh), color= (255,0,0), thickness=3 )

        # preprocessing steps
        eye= frame[ey:ey+eh,ex:ex+w]
        eye= cv2.resize(eye,(80,80))
        eye= eye/255
        eye= eye.reshape(80,80,3)
        eye= np.expand_dims(eye,axis=0)
```

Results and Graphs

Open eyes



Closed eyes



Conclusion

In this proposed work a new method is proposed for driver drowsiness detection based on eye state. This determines the state of the eye that is drowsy or non- drowsy and alert with an alarm when state of the eye is drowsy. Face and eye region are detected using Predict and Detection algorithm. Stacked deep convolution neural network is developed to extract features and used for learning phase. An EAR equation is used to classify the driver as sleep or non-sleep. Proposed system achieved (92%>) accuracy. Proposed system effectively identifies the state of driver and alert with an alarm, when the model predicts drowsy output state continuously. We have used transfer learning to improve the performance of the system. By doing this many accidents will be reduced and provide safe life to driver and vehicle safety. A system for car safety and driver safety is presented only in the luxurious cars. Using drowsiness detection system, driver safety can be implemented in normal cars also.

References

1. S. E. Viswapriya , Singamsetti Balabalaji , Yedida Sireesha (2021) A Machine-Learning Approach for Driver-Drowsiness Detection based on Eye-State, Volume 10 Issue 4
2. Marnim Galib (2019) A Realistic Dataset and Baseline Temporal Model for Early Drowsiness Detection
3. T. Vesselenyi (2017) Driver drowsiness detection using ANN image processing IOP Conf. Series: Materials Science and Engineering 252 (2017) 012097
4. Rateb Jabbara*, Khalifa Al-Khalifaa , Mohamed Kharbechea , Wael Alhajyaseena , Mohsen Jafarib , Shan Jiang (2015) Real-time Driver Drowsiness Detection for Android Application Using Deep Neural Networks Techniques, The 9th International Conference on Ambient Systems, Networks, and Technologies (ANT 2018)
5. MONAGI H. ALKINANI 1 , WAZIR ZADA KHAN 2 , (Senior Member, IEEE), AND QURATULAIN ARSHAD (2020), Detecting Human Driver Inattentive and Aggressive Driving Behavior Using Deep Learning: Recent Advances, Requirements and Open Challenges, Saudi Arabia 2Farasan Networking Research Laboratory

Appendix (code)

Appendix I
Output Video Link
https://drive.google.com/file/d/1oreyDs0GtkGBwj32HIFV7NDSFLPnjTPJ/view

Appendix II
Github Link
https://github.com/DarshGupta1910/Driver_Drowsiness_Detection_using_Machine_Learning

Plagiarism report

Number of Words : 935

Results Found : 24

To or From

Binary Translator

To or From

PDF Converter

10%

Plagiarism

90%

Unique

Make it Unique

Start New Search

To check plagiarism in photos click here

Reverse Image Search

Number of Words : 775

Results Found : 16

To or From

Binary Translator

To or From

PDF Converter

12%

Plagiarism

88%

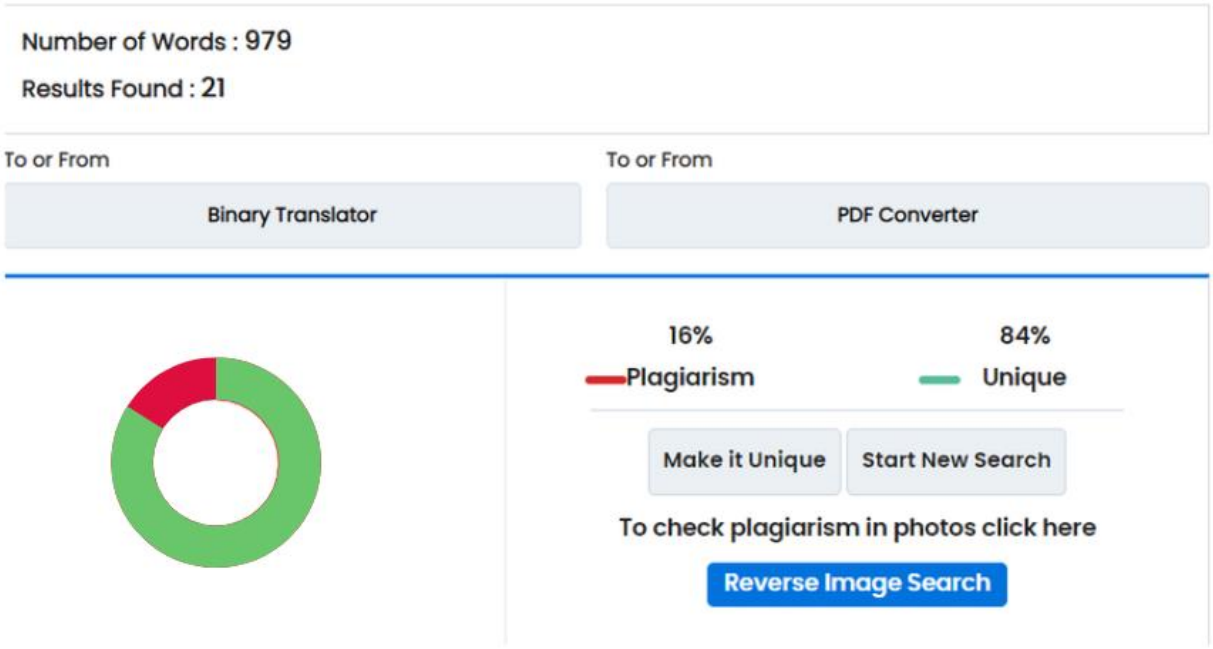
Unique

Make it Unique

Start New Search

To check plagiarism in photos click here

Reverse Image Search



Taking average of above 3 results we get

Plagiarism = 12.67%
Unique = 87.34%