Task-0

Driver Drowsiness Detection System

Ishu Khandelwal

Date-06/06/2022

Abstract

The goal of this effort is to build an ADAS (advanced driving assistance system) focusing on driver drowsiness detection, with the goal of alerting drivers of their fatigued condition in order to avert road traffic accidents. It is critical in a driving context that tiredness detection is non-intrusive and that the driver is not distracted by alerts when he or she is not sleepy. Our solution to this unsolved problem consists of 60-second picture sequences that are captured in such a way that the subject's face is visible combining a wristwatch that measures the inputs like a heartbeat, blood pressure, heart rate, body temperature, and ECG and produces the output. The combination of the output of both these gives a solution. A solution is being developed to determine whether the driver is drowsy or not, with the goal of minimizing false positives. Despite the fact that the acquired findings are not particularly satisfying, the recommendations offered in this study are promising and can be used as a firm foundation for future research.

This research describes an autonomous sleepy driver detection and accident prevention system based on variations in the length of eye blinks. Using the hypothesized horizontal symmetry property of the eyes, our proposed approach detects visual changes in eye positions. Our innovative technology uses a regular webcam that is precisely positioned in front of the driver's seat to identify eye blinks. It will detect the eyeballs based on a certain EAR (Eye Aspect Ratio).

Introduction

Driver drowsiness detection is an automobile safety feature that detects when the driver is becoming drowsy and avoids accidents. According to many studies, driver weariness is a key component in a huge number of vehicle accidents, accounting for roughly 20% of all road accidents and up to 50% on particular highways. According to recent figures, fatigue-related collisions result in 1,200 deaths and 76,000 injuries per year. In the realm of accident-avoidance systems, developing technology for detecting or preventing tiredness behind the wheel is a big issue. Because of the dangers that sleepiness poses on the road, strategies for combating its effects must be devised. On certain roads, up to 50% of the traffic is connected. In a huge proportion of cases, driver weariness is a significant issue.

In a substantial proportion of car accidents, driver fatigue is a crucial element. According to recent figures, fatigue-related collisions result in 1,200 deaths and 76,000 injuries per year. In the realm of accident-avoidance systems, developing technology for detecting or preventing tiredness behind the wheel is a big issue. Because of the dangers that sleepiness poses on the road, measures for counteracting its effects must be devised. Drowsiness detection is one of the most prevalent issues that must be addressed in order to prevent traffic accidents. Automobile fatigue-related crashes have increased dramatically in recent years.

Problem Statement

Nowadays drowsiness of drivers is one of the main reasons behind road accidents. It is natural for the drivers who take long drives to doze off behind the steering wheel.

The risk, danger, and often tragic results of drowsy driving are alarming. Drowsy driving is the dangerous combination of driving and sleepiness or fatigue. This usually happens when a driver has not slept enough, but it can also happen because of untreated sleep disorders, medications, drinking alcohol, or shift work.

Driver's inattention might be the result of a lack of alertness when driving due to driver drowsiness and distraction. Driver distraction occurs when an object or event draws a person's attention away from the driving task. Unlike driver distraction, driver drowsiness involves no triggering event but, instead, is characterized by a progressive withdrawal of attention from the road and traffic demands. Both driver drowsiness and distraction, however, might have the same effects, that is decreased driving performance, longer reaction time, and an increased risk of crash involvement.
No one knows the exact moment when sleep comes over their body. Falling asleep at the wheel is clearly dangerous but being sleepy affects your ability to drive safely even if you don't fall asleep. Drowsiness:

- Makes you less able to pay attention to the road.
- Slows reaction time if you must brake or steer suddenly.
- Affects your ability to make good decisions.

Market Need Assessment

- Reliability: The solution should reliably detect drowsiness so that it can serve its purpose as a system for promoting driver safety.
- Real-time response: The operation of a vehicle can involve relatively high speeds, a system that cannot detect drowsiness and warn that driver promptly can lead to serious consequences.

- Unobtrusive: It is very important that the solution is as transparent to the driver as possible.

Customer Need Assessment

- **Provides alerts/warnings for driver fatigue and distraction:** Generates alarms to avoid accidents caused by distractions while driving like talking to passengers, bowing the head, gazing around, or spending a considerable amount of time in operating audio and navigation systems.
- **Excellent detection ability even with glasses:** With regular glasses or non-reflection light sunglasses, the drowsiness detection system can always make an accurate analysis of the driver's fatigue state.
- Warns the driver of drowsiness and the risk of a microsleep.
- Compliance with driver warnings helps to avoid crashes caused by fatigue

Business Assessment

- **Easy to be installed and used:**
  Our edge computing device has a small and compact size, which hardly influences the driver's view of sight. It can be easily installed on the dashboard.
- **Works around all day, under any weather:**
  Whether in the daytime or night, on rainy or snowy days, the system can always make an accurate analysis of the driver's fatigue state.

Target Specifications and Characterization

- ✓ The use of a drowsiness detection system is required.
- ✓ It will keep the driver safe while driving and will remind him to do so.

- ✓ There will be no danger of death or an accident.
- ✓ The driver will feel safe when driving at night.
- ✓ The driver will be notified.

External Search

The following are the sources I utilised as a reference for examining the necessity for such a system for local businesses and how companies have been utilising the strategy to increase sales:

- https://www.bosch-mobility-solutions.com/en/solutions/assistance-systems/driver-drowsiness-detection/
- https://techvidvan.com/tutorials/driver-drowsiness-detection-system/
- https://www.irjet.net/archives/V7/i6/IRJET-V7I61208.pdf
- https://ieeexplore.ieee.org/document/8704263
- https://medium.com/ai-techsystems/driver-drowsiness-detection-using-cnn-ac66863718d

Bench marking alternate products

There are many techniques available for approaching this problem and each has its sets of pros and cons. We are gonna list some and then infer why we chose one for our use.

EEG based

Electroencephalogram (EEG) signal of driver measured with a single electrode neuro-signal acquisition device and a fatigue index or activity level calculated which in turn tells if the driver is likely to fall asleep. This is a wearable type system. It is fairly robust but has to be specifically tuned for different individuals in some cases.

Steering mounted

It works by recording the steering behavior of the driver at different points in the trip and inferring the level of activity of the driver. It is often used in conjunction with monitoring different behaviors such as pressure on the acceleration pedal, movement of the car, etc.

Yawning based

A video feed of the driver's face is continuously scanned for gestures such as yawning which indicates the fatigue state of the driver. A small camera is placed inside the vehicle which records the behavior of the driver and the feed is either locally or over a server scanned for yawning and such behavior.

There are various methods like detecting objects which are near to vehicle and front and rear cameras for detecting vehicles approaching near to vehicle and airbag system which can save lives after an accident is accorded.

Disadvantages:

Most of the existing systems use external factors and inform the user about the problem and save users after an accident is accord but from research most of the accidents are due to faults in users like drowsiness, sleeping while driving

Applicable Patents

Similar to yawning-based method a camera records the driver's face and checks if the driver's eyes are closed or open. In previous products there is only basically detection of the face. The estimated average blinking duration is between 100-400ms according to the Harvard database of useful biological numbers. If the camera sees that the eye of the driver closes for a duration far more than that it marks the driver as asleep and some sort of alarm raised.

This method has proven to be the most effective measure and easiest to implement providing satisfactory results.

To deal with driver drowsiness problems and provide an effective system a drowsiness detection system can be developed with a wrist watch that can be placed in the driver's hand which will take Physiological parameters as input and compare it with training data and if the driver is showing any symptoms of drowsiness system will automatically detect and raise an alarm which will alert the driver and other passengers.

Applicable Regulations

Geographically, the automotive driver drowsiness detection system market has been segmented into North America, Europe, Asia Pacific, and the rest of the world. North America holds the largest share of the global driver drowsiness detection system market owing to the high volume of commercial vehicles in this region. The government has also imposed stringent regulations regarding road safety and made it mandatory for all the automotive manufacturing companies to install the DDDS in every vehicle. Moreover, the increasing adoption of advanced technologies in vehicles is propelling the growth of the market in this region. The Asia Pacific is also expected to show a significant growth in the DDDS market owing to the increasing number of sales and production of vehicles in this region. Some of the developing economies such as India, China, and Japan are rapidly adopting advanced technology to develop safety systems in vehicles. Also, the increasing inclination of people towards the enhanced safety features in vehicles is fuelling the growth of the DDDS market in this region.

Applicable Constraints

- High costs are expected to hinder the growth of the heads-up display market.
- One of the challenges in developing an efficient drowsiness detection system is how to obtain proper drowsiness data. Due to safety reasons, drowsiness cannot be manipulated in a real environment; thus, the drowsiness detection system has to be developed and tested in a laboratory setting.
- At the night, it is impossible to capture a photo of the driver so the detection of drowsiness will affect.

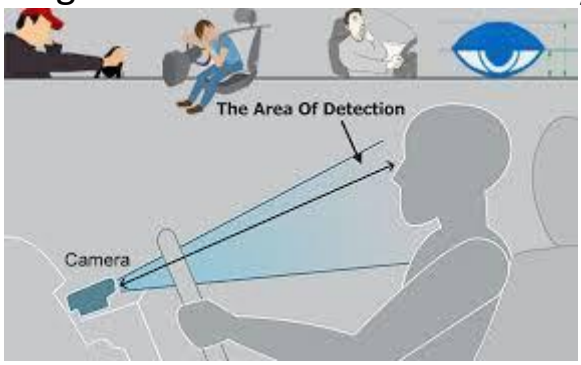
Business Opportunity

A car accident is the major cause of death in which around 1.3 million people die every year. The whole system is deployed on portable hardware which can be easily installed in the car for use. As this technology helps to prevent car accidents due to drowsiness so people will buy this product.

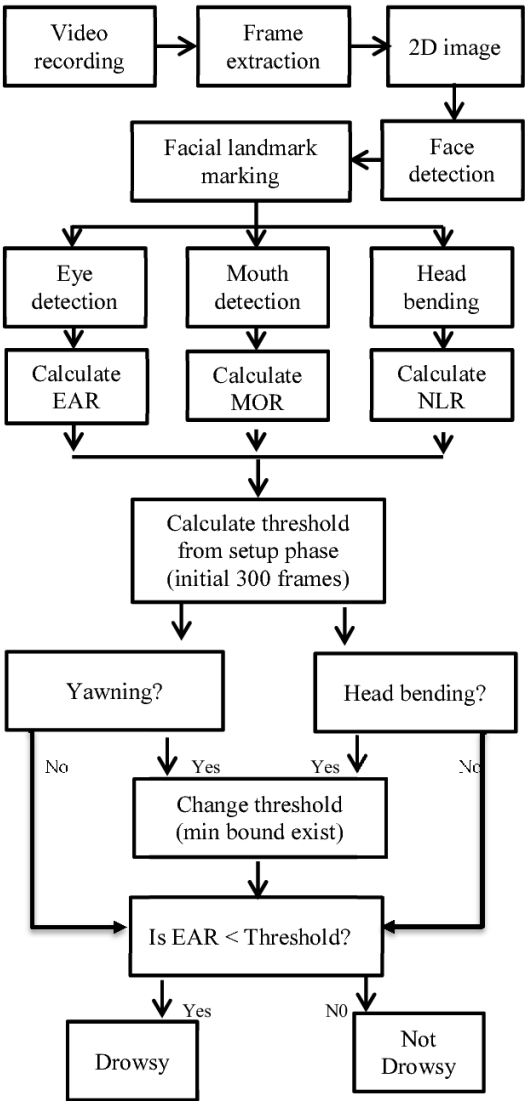
Code Generation

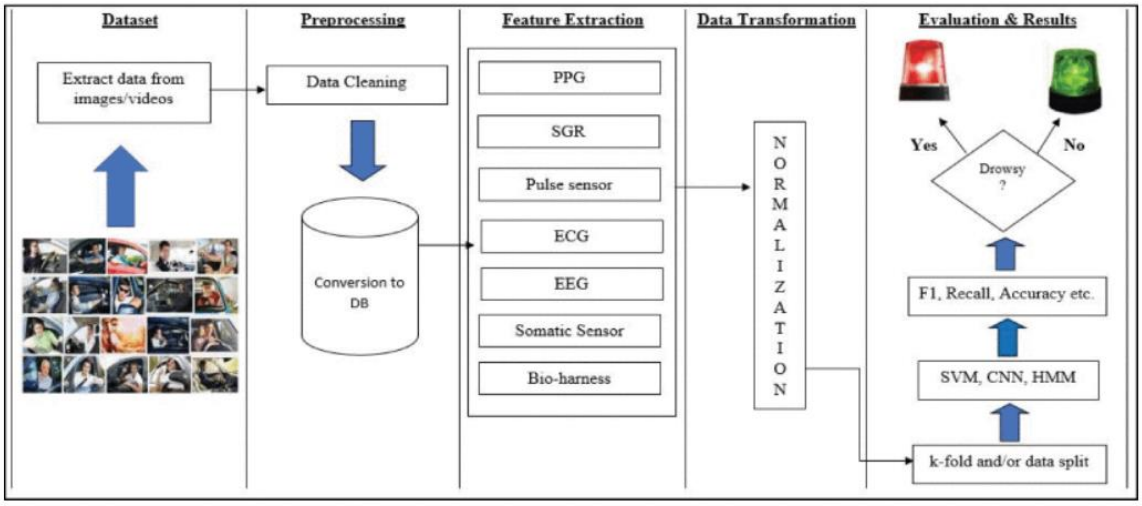
A countless number of people drive long-distance every day and night on the highway. Lack of sleep or distractions like the phone call, talking with the passenger, etc may lead to an accident.

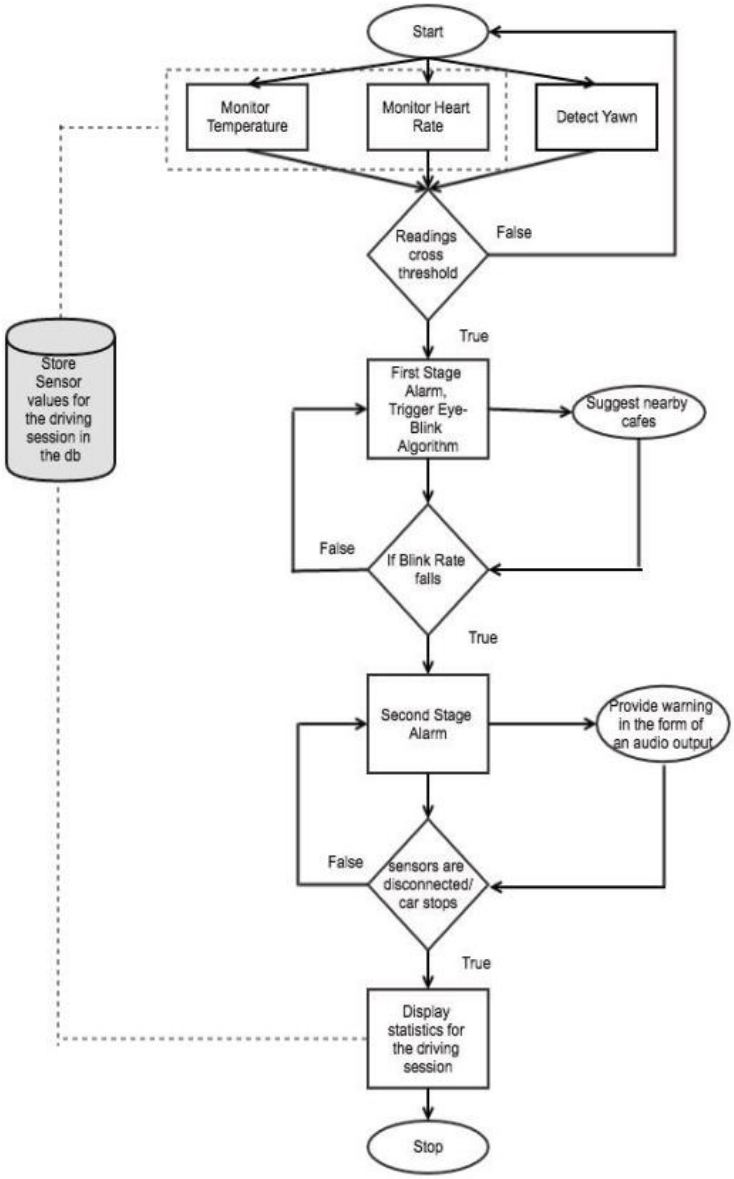
To prevent such accidents, we propose a system that alerts the driver if the driver gets distracted or feels drowsy.



Final Prototype




Dataset
Preprocessing
Feature Extraction
Data Transformation
Evaluation & Results
Extract data from images/videos
Data Cleaning
Conversion to DB
PPG
SGR
Pulse sensor
ECG
EEG
Somatic Sensor
Bio-harness
NORMALIZATION
Yes
No
Drowsy ?
F1, Recall, Accuracy etc.
SVM, CNN, HMM
k-fold and/or data split


Start
Monitor Temperature
Monitor Heart Rate
Detect Yawn
Readings cross threshold
False
True
Store Sensor values for the driving session in the db
First Stage Alarm, Trigger Eye-Blink Algorithm
Suggest nearby cafes
False
If Blink Rate falls
True
Second Stage Alarm
Provide warning in the form of an audio output
False
sensors are disconnected/ car stops
True
Display statistics for the driving session
Stop

Product details

HARDWARE REQUIREMENTS

- **Camera-** The camera quality should be good enough to record the video and capture images.
- **Personal computer - The** laptop being used should be updated to save the video and images. The laptop should be able to give a preview of images, plots, and videos

SOFTWARE REQUIREMENTS

- **Python 2.7 or above versions**
- **Anaconda software**

Cost

The cost of making the product will be in the range of Rs 10,000 – 25,000 depending on how efficient the product the customer wants.

Data Visualization

#Import the necessary libraries.

```
[1]: import numpy as np
     import pandas as pd
     import os
     import cv2
```

```
[2]: labels = os.listdir("dataset")
```

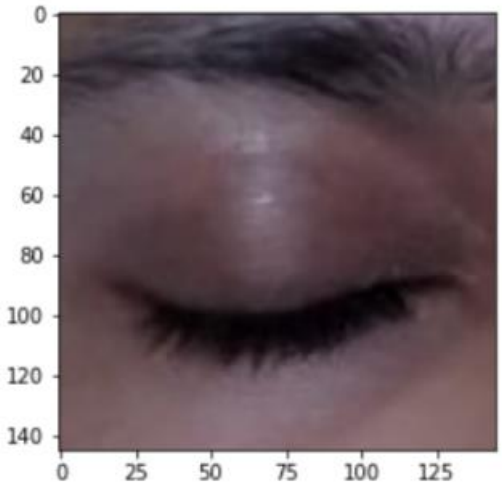
```
[3]: labels
```

```
[3]: labels
```

```
[3]: ['Closed', 'no_yawn', 'Open', 'yawn']
```

[4]:
```
import matplotlib.pyplot as plt
plt.imshow(plt.imread("dataset/Closed/_0.jpg"))
```

[4]:
```
<matplotlib.image.AxesImage at 0x1ff76b46610>
```



[5]:
```
a = plt.imread("dataset/yawn/10.jpg")
```

[6]:
```
a.shape
```

[6]:
```
(480, 640, 3)
```

visualize yan image.

[7]:
```
plt.imshow(plt.imread("dataset/yawn/10.jpg"))
```

[7]:
```
<matplotlib.image.AxesImage at 0x1ff76bf8940>
```

[8]:
```
def face_for_yawn(direc="dataset", face_cas_path="haarcascade_frontalface_default.xml"):
    yaw_no = []
    IMG_SIZE = 145
    categories = ["yawn", "no_yawn"]
    for category in categories:
        path_link = os.path.join(direc, category)
        class_num1 = categories.index(category)
        print(class_num1)
        for image in os.listdir(path_link):
            image_array = cv2.imread(os.path.join(path_link, image), cv2.IMREAD_COLOR)
            face_cascade = cv2.CascadeClassifier(face_cas_path)
            faces = face_cascade.detectMultiScale(image_array, 1.3, 5)
            for (x, y, w, h) in faces:
                img = cv2.rectangle(image_array, (x, y), (x+w, y+h), (0, 255, 0), 2)
                roi_color = img[y:y+h, x:x+w]
                resized_array = cv2.resize(roi_color, (IMG_SIZE, IMG_SIZE))
                yaw_no.append([resized_array, class_num1])
    return yaw_no


yawn_no_yawn = face_for_yawn()
```

```
0
1
```

[9]:
```
def get_data(dir_path="dataset/", face_cas="haarcascade_frontalface_default.xml", eye_cas="haarcascade_eye.xml"):
    labels = ['Closed', 'Open']
    IMG_SIZE = 145
    data = []
    for label in labels:
        path = os.path.join(dir_path, label)
        class_num = labels.index(label)
        class_num +=2
        print(class_num)
        for img in os.listdir(path):
            try:
                img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_COLOR)
                resized_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
                data.append([resized_array, class_num])
            except Exception as e:
                print(e)
    return data
```

[10]:
```
data_train = get_data()
```

```
2
3
```

[11]:
```
def append_data():
#     total_data = []
    yaw_no = face_for_yawn()
    data = get_data()
    yaw_no.extend(data)
    return np.array(yaw_no)
```

[12]:
```
new_data = append_data()
```

```
0
1
2
3
```

[13]:
```
X = []
y = []
for feature, label in new_data:
    X.append(feature)
    y.append(label)
```

14]:
```
X = np.array(X)
X = X.reshape(-1, 145, 145, 3)
```

[15]:
```
from sklearn.preprocessing import LabelBinarizer
label_bin = LabelBinarizer()
y = label_bin.fit_transform(y)
```

[16]:
```
y = np.array(y)
```

[17]:
```
from sklearn.model_selection import train_test_split
seed = 42
test_size = 0.30
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=seed, test_size=test_size)
```

[18]:
```
len(X_test)
```

[18]:
```
578
```

[19]:
```
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten, Conv2D, MaxPooling2D, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.models import Sequential
from keras.preprocessing.image import ImageDataGenerator
import tensorflow as tf
```

[20]:
```
tf.__version__
```

[20]:
```
'2.9.1'
```

[21]:
```
import keras
keras.__version__
```

[21]:
```
'2.9.0'
```

[22]:
```
train_generator = ImageDataGenerator(rescale=1/255, zoom_range=0.2, horizontal_flip=True, rotation_range=30)
test_generator = ImageDataGenerator(rescale=1/255)

train_generator = train_generator.flow(np.array(X_train), y_train, shuffle=False)
test_generator = test_generator.flow(np.array(X_test), y_test, shuffle=False)
```

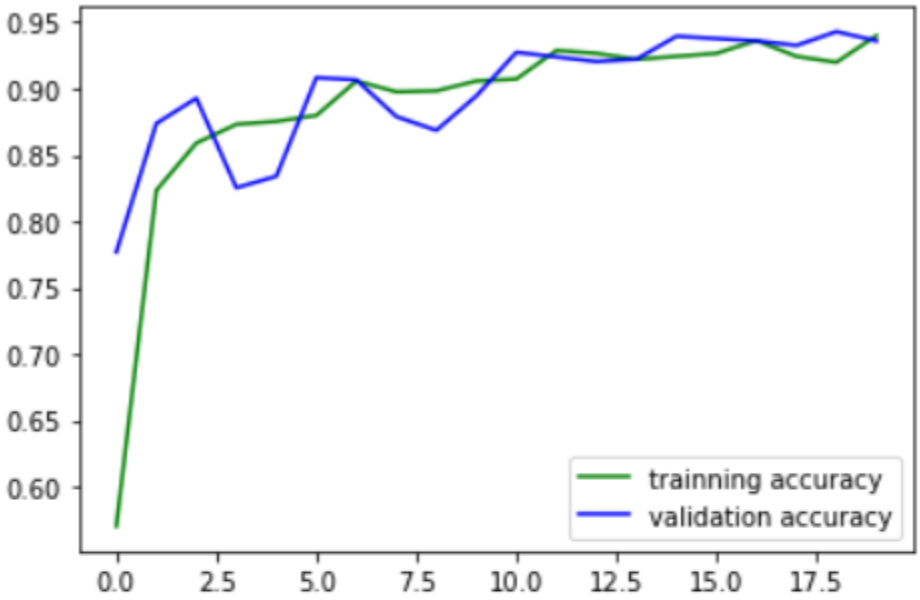
[23]:
```
model = Sequential()

model.add(Conv2D(256, (3, 3), activation="relu", input_shape=X_train.shape[1:]))
model.add(MaxPooling2D(2, 2))

model.add(Conv2D(128, (3, 3), activation="relu"))
model.add(MaxPooling2D(2, 2))

model.add(Conv2D(64, (3, 3), activation="relu"))
model.add(MaxPooling2D(2, 2))

model.add(Conv2D(32, (3, 3), activation="relu"))
model.add(MaxPooling2D(2, 2))

model.add(Flatten())
model.add(Dropout(0.5))

model.add(Dense(64, activation="relu"))
model.add(Dense(4, activation="softmax"))

model.compile(loss="categorical_crossentropy", metrics=["accuracy"], optimizer="adam")

model.summary()
```

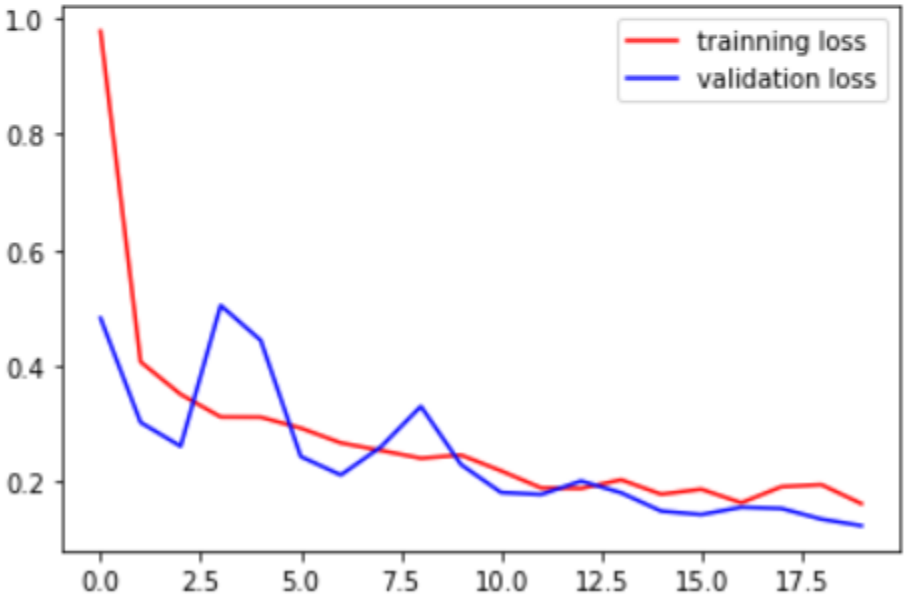
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 143, 143, 256)     7168

 max_pooling2d (MaxPooling2D  (None, 71, 71, 256)      0
 )

 conv2d_1 (Conv2D)           (None, 69, 69, 128)       295040

 max_pooling2d_1 (MaxPooling  (None, 34, 34, 128)      0
 2D)

 conv2d_2 (Conv2D)           (None, 32, 32, 64)        73792

 max_pooling2d_2 (MaxPooling  (None, 16, 16, 64)       0
 2D)

 conv2d_3 (Conv2D)           (None, 14, 14, 32)        18464

 max_pooling2d_3 (MaxPooling  (None, 7, 7, 32)         0
 2D)

 flatten (Flatten)           (None, 1568)              0

 dropout (Dropout)           (None, 1568)              0

 dense (Dense)               (None, 64)                100416

 dense_1 (Dense)             (None, 4)                 260

=================================================================
Total params: 495,140
Trainable params: 495,140
Non-trainable params: 0
```

[24]:
```
history = model.fit(train_generator, epochs=20, validation_data=test_generator, shuffle=True, validation_steps=len(test_generator))
print(history)
```

```
Epoch 1/20
43/43 [==============================] - 205s 5s/step - loss: 0.9782 - accuracy: 0.5702 - val_loss: 0.4820 - val_accuracy: 0.7768
Epoch 2/20
43/43 [==============================] - 347s 8s/step - loss: 0.4064 - accuracy: 0.8233 - val_loss: 0.3017 - val_accuracy: 0.8737
Epoch 3/20
43/43 [==============================] - 181s 4s/step - loss: 0.3503 - accuracy: 0.8589 - val_loss: 0.2604 - val_accuracy: 0.8927
Epoch 4/20
43/43 [==============================] - 197s 5s/step - loss: 0.3111 - accuracy: 0.8731 - val_loss: 0.5040 - val_accuracy: 0.8253
Epoch 5/20
43/43 [==============================] - 185s 4s/step - loss: 0.3108 - accuracy: 0.8753 - val_loss: 0.4436 - val_accuracy: 0.8339
Epoch 6/20
43/43 [==============================] - 190s 4s/step - loss: 0.2918 - accuracy: 0.8797 - val_loss: 0.2427 - val_accuracy: 0.9083
Epoch 7/20
43/43 [==============================] - 186s 4s/step - loss: 0.2663 - accuracy: 0.9057 - val_loss: 0.2107 - val_accuracy: 0.9066
Epoch 8/20
43/43 [==============================] - 469s 11s/step - loss: 0.2532 - accuracy: 0.8976 - val_loss: 0.2593 - val_accuracy: 0.8789
Epoch 9/20
43/43 [==============================] - 233s 5s/step - loss: 0.2397 - accuracy: 0.8983 - val_loss: 0.3294 - val_accuracy: 0.8685
Epoch 10/20
43/43 [==============================] - 208s 5s/step - loss: 0.2453 - accuracy: 0.9057 - val_loss: 0.2289 - val_accuracy: 0.8945
Epoch 11/20
43/43 [==============================] - 193s 4s/step - loss: 0.2180 - accuracy: 0.9072 - val_loss: 0.1806 - val_accuracy: 0.9273
Epoch 12/20
43/43 [==============================] - 197s 5s/step - loss: 0.1885 - accuracy: 0.9287 - val_loss: 0.1770 - val_accuracy: 0.9239
Epoch 13/20
43/43 [==============================] - 187s 4s/step - loss: 0.1876 - accuracy: 0.9265 - val_loss: 0.2003 - val_accuracy: 0.9204
Epoch 14/20
43/43 [==============================] - 180s 4s/step - loss: 0.2025 - accuracy: 0.9220 - val_loss: 0.1800 - val_accuracy: 0.9221
Epoch 15/20
43/43 [==============================] - 173s 4s/step - loss: 0.1774 - accuracy: 0.9243 - val_loss: 0.1483 - val_accuracy: 0.9394
Epoch 16/20
43/43 [==============================] - 159s 4s/step - loss: 0.1862 - accuracy: 0.9265 - val_loss: 0.1423 - val_accuracy: 0.9377
Epoch 17/20
43/43 [==============================] - 152s 4s/step - loss: 0.1627 - accuracy: 0.9362 - val_loss: 0.1551 - val_accuracy: 0.9360
```

[25]:
```
accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(len(accuracy))

plt.plot(epochs, accuracy, "g", label="trainning accuracy")
plt.plot(epochs, val_accuracy, "b", label="validation accuracy")
plt.legend()
plt.show()

plt.plot(epochs, loss, "r", label="trainning loss")
plt.plot(epochs, val_loss, "b", label="validation loss")
plt.legend()
plt.show()
```





[26]:
```
model.save("drowiness_new20.h5")
```

[31]:
```
model.save("drowiness_new20.model")
```

[33]:
```
prediction = (model.predict(X_test) > 0.5).astype("int32")
```

```
19/19 [==============================] - 17s 806ms/step
```

[34]:
```
prediction
```

[34]:
```
array([[0, 0, 0, 1],
       [0, 0, 0, 1],
       [0, 0, 1, 0],
       ...,
       [0, 0, 0, 1],
       [0, 0, 1, 0],
       [0, 0, 1, 0]])
```

[35]:
```
labels_new = ["yawn", "no_yawn", "Closed", "Open"]
```

[40]:
```
from sklearn.metrics import confusion_matrix
y_test_arg=np.argmax(y_test,axis=1)
Y_pred = np.argmax(model.predict(X_test),axis=1)
print('Confusion Matrix')
print(confusion_matrix(y_test_arg, Y_pred))
```

```
19/19 [==============================] - 17s 873ms/step
Confusion Matrix
[[ 47  13   3   0]
 [  1  73   0   0]
 [  1   3 208   3]
 [  1   0  23 202]]
```

[41]:
```
labels_new = ["yawn", "no_yawn", "Closed", "Open"]
IMG_SIZE = 145
def prepare(filepath, face_cas="haarcascade_frontalface_default.xml"):
    img_array = cv2.imread(filepath, cv2.IMREAD_COLOR)
    img_array = img_array / 255
    resized_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
    return resized_array.reshape(-1, IMG_SIZE, IMG_SIZE, 3)

model = tf.keras.models.load_model("./drowiness_new20.h5")
```

[42]:
```
prediction = model.predict([prepare("dataset/no_yawn/1067.jpg")])
np.argmax(prediction)
```

```
1/1 [==============================] - 0s 331ms/step
```

[42]:
```
2
```

Code implementation

The libraries need for driver drowsiness detection system are

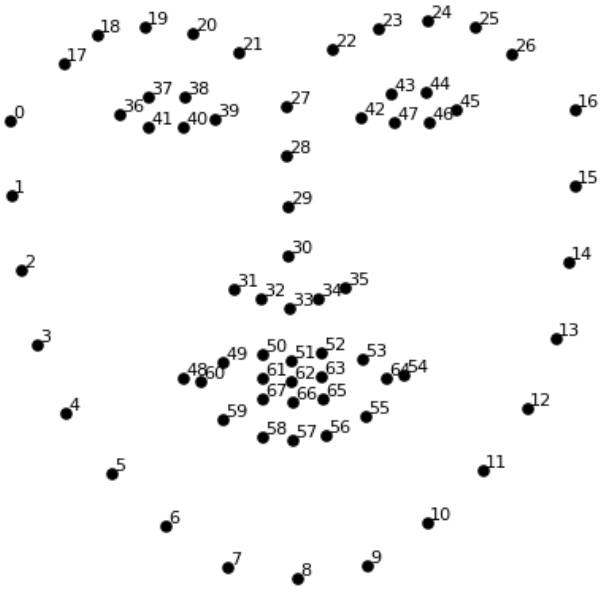
- Opencv
- Dlib
- Numpy

Numpy is used for handling the data from dlib and mathematical functions. Opencv will help us in gathering the frames from the webcam and writing over them and also displaying the resultant frames.

Dlib to extract features from the face and predict the landmark using its pre-trained face landmark detector.

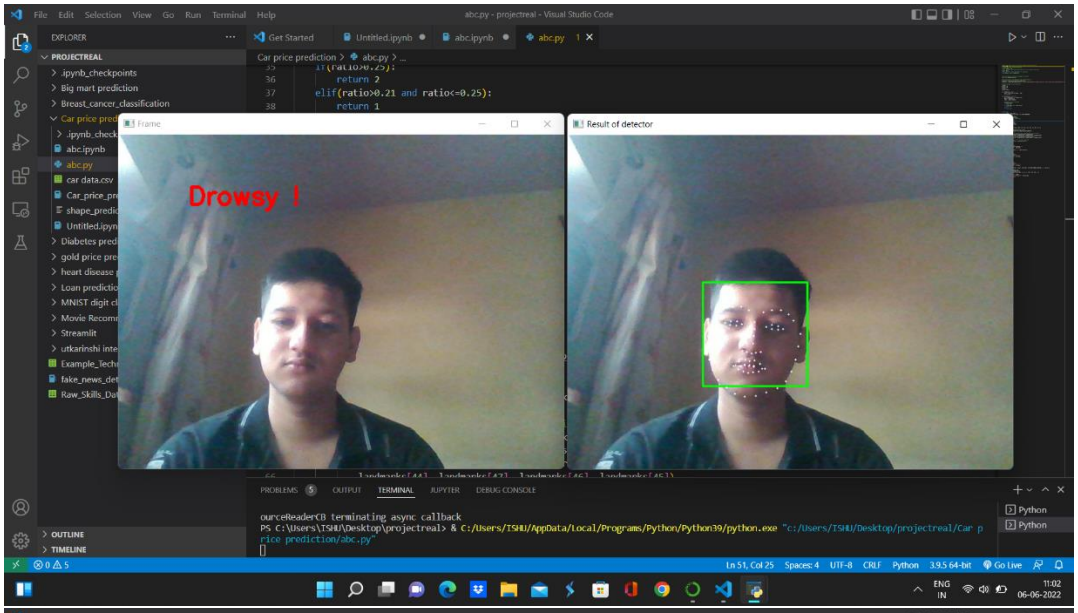
Dlib is an open source toolkit written in C++ that has a variety of machine learning models implemented and optimized. Preference is given to dlib over other libraries and training your own model because it is fairly accurate, fast, well documented, and available for academic, research, and even commercial use.

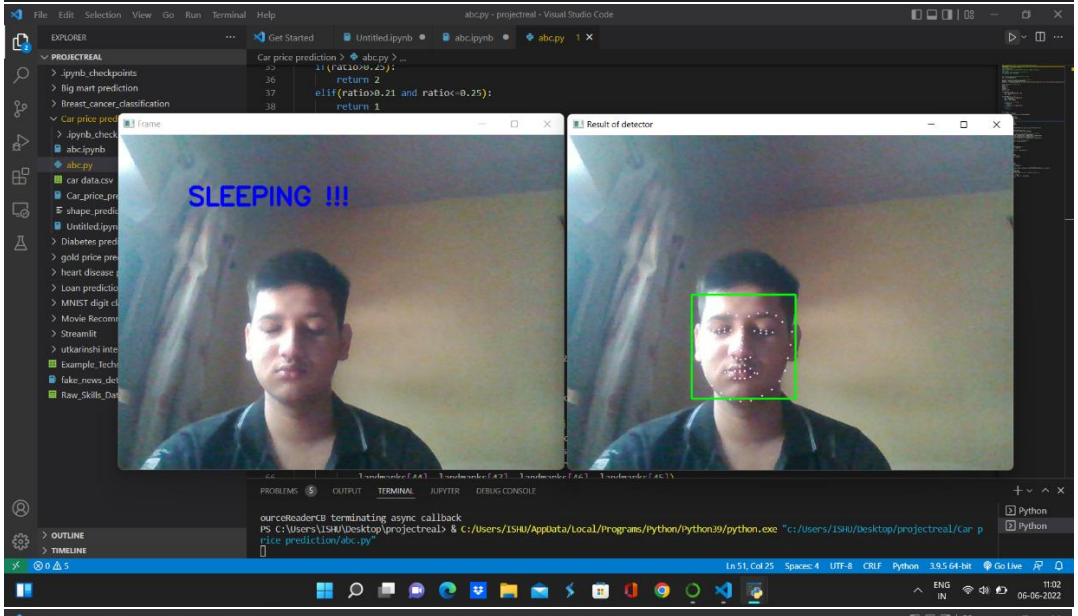
Dlib's accuracy and speed are comparable with the most state-of-the-art neural networks, and because the scope of this project is not to train one, we'll be using dlib python wrapper.

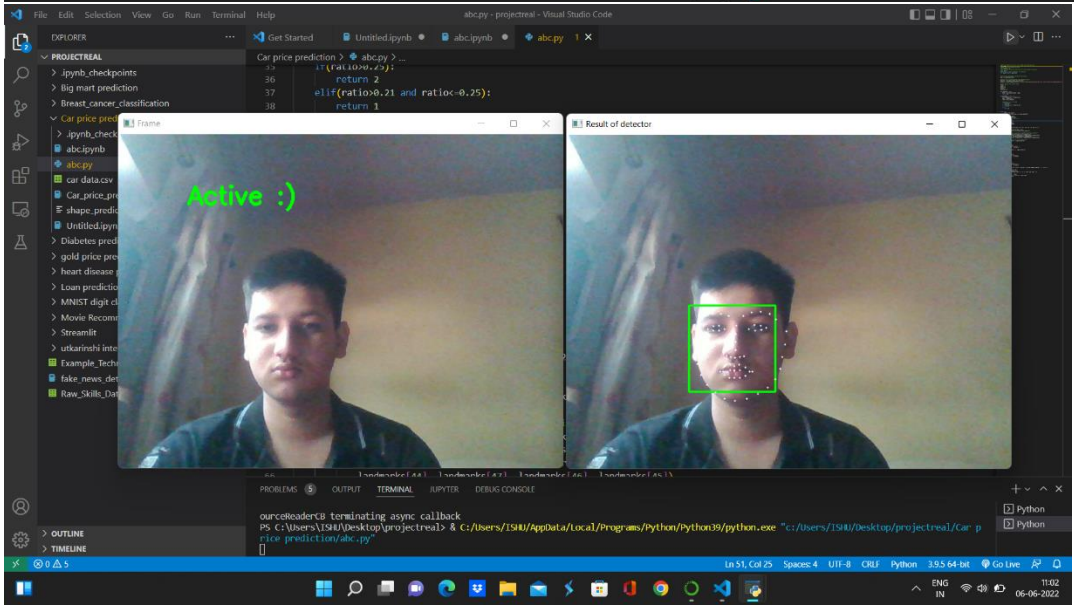


```
import cv2
import numpy as np
import dlib
#face_utils for basic operations of conversion
from imutils import face_utils

cap = cv2.VideoCapture(0)
#Initializing the face detector and landmark detector
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor("C:\\Users\\ISHU\\Desktop\\projectreal\\Car
price prediction\\shape_predictor_68_face_landmarks.dat ")

sleep = 0
drowsy = 0
active = 0
status=""
color=(0,0,0)

def compute(ptA,ptB):
    dist = np.linalg.norm(ptA - ptB)
    return dist

def blinked(a,b,c,d,e,f):
    up = compute(b,d) + compute(c,e)
    down = compute(a,f)
    ratio = up/(2.0*down)

    #Checking if it is blinked
    if(ratio>0.25):
        return 2
    elif(ratio>0.21 and ratio<=0.25):
        return 1
    else:
        return 0
while True:
    _, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = detector(gray)
    face_frame = frame.copy()
    #detected face in faces array
    for face in faces:
        x1 = face.left()
        y1 = face.top()
        x2 = face.right()
        y2 = face.bottom()
```

```
        cv2.rectangle(face_frame, (x1, y1), (x2, y2), (0, 255, 0), 2)

        landmarks = predictor(gray, face)
        landmarks = face_utils.shape_to_np(landmarks)

        #The numbers are actually the landmarks which will show eye
        left_blink = blinked(landmarks[36],landmarks[37],
            landmarks[38], landmarks[41], landmarks[40], landmarks[39])
        right_blink = blinked(landmarks[42],landmarks[43],
            landmarks[44], landmarks[47], landmarks[46], landmarks[45])

        #Now judge what to do for the eye blinks
        if(left_blink==0 or right_blink==0):
            sleep+=1
            drowsy=0
            active=0
            if(sleep>6):
                status="SLEEPING !!!"
                color = (255,0,0)

        elif(left_blink==1 or right_blink==1):
            sleep=0
            active=0
            drowsy+=1
            if(drowsy>6):
                status="Drowsy !"
                color = (0,0,255)

        else:
            drowsy=0
            sleep=0
            active+=1
            if(active>6):
                status="Active :)"
                color = (0,255,0)

        cv2.putText(frame, status, (100,100), cv2.FONT_HERSHEY_SIMPLEX, 1.2,
color,3)

        for n in range(0, 68):
            (x,y) = landmarks[n]
            cv2.circle(face_frame, (x, y), 1, (255, 255, 255), -1)

    cv2.imshow("Frame", frame)
    cv2.imshow("Result of detector", face_frame)
    key = cv2.waitKey(1)
    if key == 27:
          break
```


Drowsy !
Result of detector


SLEEPING !!!
Result of detector


Active :)
Result of detector

CONCLUSION

In order to detect a driver's drowsiness, facial features, eyes, and mouth were identified on the video of the individual driving. The algorithm is implemented to classify eyes as open or closed. Wrist Watch is put in the driver's hands. Drowsiness was determined on the basis of the frequency of closed eyes and biological inputs. Using OpenCV and Dlib in Python, the frequency of yawning was examined. An alarm was set to ring after the detection to alert the driver. There will be limitations concerning the detection of drivers' conditions and facial expressions due to factors like darkness, light reflection, obstructions by drivers' hands, and wearing of sunglasses. Convolutional neural gives better performance and the facial extraction method accompanies it, as an additional drowsiness detection technique which is often used with other facial extraction methods.