AISSMS
INSTITUTE OF INFORMATION TECHNOLOGY
ADDING VALUE TO ENGINEERING

Department Of Computer Engineering

A PROJECT BASED LEARING REPORT ON

DRIVER DROWSINESS SYSTEM

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING
AISSMS IOIT

SE COMPTER ENGINEERING

SUBMITTED BY

STUDENT NAME	ERP No:
Onasvee Banarse	09
Himanshu Bendale	10
Kaustubh Kabra	34
Harsh Shah	59
Vincent Simon	75

2020 -2021

AISSMS
INSTITUTE OF INFORMATION TECHNOLOGY
ADDING VALUE TO ENGINEERING

Department of Computer Engineering

CERTIFICATE

This is to certify that the project report
"DRIVER DROWSINESS SYSTEM"
Submitted by

STUDENT NAME	ERP No:
Onasvee Banarse	09
Himanshu Bendale	10
Kaustubh Kabra	34
Harsh Shah	59
Vincent Simon	75

is a bonafide student of this institute and the work has been carried out by him/her under the supervision of **Prof. Anuradha Varal** and it is approved for the partial fulfillment of the Department of Computer Engineering AISSMS IOIT.

(Prof. Anuradha Varal)
Guide
Department of Computer Engineering

(Dr. S. N. Zaware)
Head,
Department of Computer Engineering

Place: Pune

Date: 20/12/2020

ABSTRACT

Driver fatigue has become one of the key reasons for road accidents in modern days. Various surveys prove that if a driver is correctly identified as fatigued, and he, or she is timely alarmed regarding the same, the cases of accidents can be remarkably reduced. There have been various techniques adopted to identify a drowsy driver. Through this project, an in-depth study of various existing techniques of fatigue in a driver is studied, followed by developing a deep learning based model to accurately identify a driver's state using a novel technique of using spatiotemporal features of the face. It can be determined that the accuracy will remarkably increase when this technique is used.

Drowsy Driver Detection System has been developed using a non-intrusive machine vision based concepts. The system uses a small monochrome security camera that points directly towards the driver's face and monitors the driver's eyes in order to detect fatigue. In such a case when fatigue is detected, a warning signal is issued to alert the driver. This report describes how to find the eyes, and also how to determine if the eyes are open or closed. The algorithm developed is unique to any currently published papers, which was a primary objective of the project. The system deals with using information obtained for the binary version of the image to find the edges of the face, which narrows the area of where the eyes may exist. Once the face area is found, the eyes are found by computing the horizontal averages in the area. Taking into account the knowledge that eye regions in the face present great intensity changes, the eyes are located by finding the significant intensity changes in the face. Once the eyes are located, measuring the distances between the intensity changes in the eye area determine whether the eyes are open or closed. A large distance corresponds to eye closure. If the eyes are found closed for 5 consecutive frames, the system draws the conclusion that the driver is falling asleep and issues a warning signal. The system is also able to detect when the eyes cannot be found, and works under reasonable lighting conditions.

TABLE OF CONTENTS

1.PROBLEM STATEMENT

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.

Develop an application for Detection of Drowsiness using Eye Tracking.

2. OBJECTIVE

1. To develop safety sytem for drivers who travel more and during night time.
2. To achieve neural network model to analyze

3 DESCRIPTION

3.1 EXISTING METHODS OF DROWSINESS DETECTION:

Driver fatigue detection methods are implemented via mathematical models, shallow models or deep models.

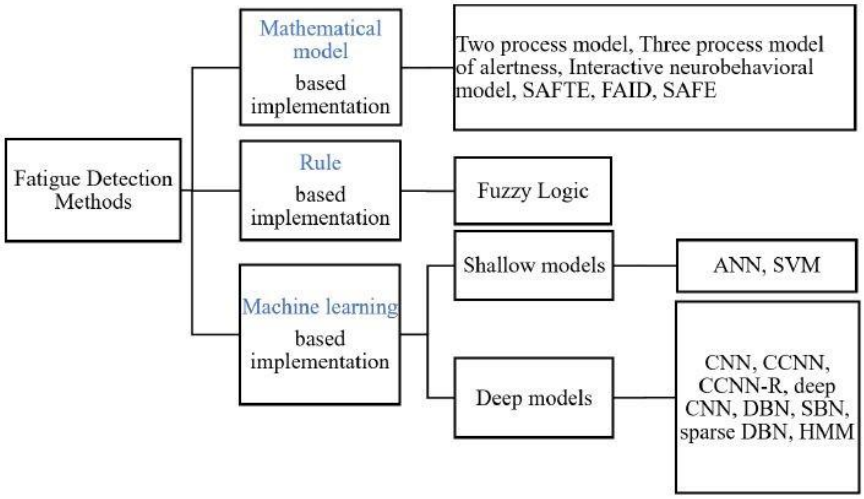


Figure. 1. *Fatigue Detection Methods*

3.1.1 MATHEMATICAL MODELS:

Bio-mathematical models offer a quantitative analysis of the effect of sleep cycle on individual performance. Types of inputs used typically are circadian cycles, duration of sleep, duration of wakefulness and sleep history to predict risk of fatigue and performance quality.

One of the earliest models is the Two Process Model. This model is based on the interaction of two processes, i.e., the circadian Process 'C' and the homeostatic Process 'S'. These processes predict performance and fatigue levels. An upgrade to the two process model is the Three Process Model of Alertness, which utilizes the duration of sleep and wakefulness as input to predict fatigue risk and alertness. This computer based model considers both circadian and homeostatic components.

Various other models are developed in continuation of this which incorporate various combinations of possible inputs to predict fatigue.

3.1.2 RULE BASED MODELS:

Rule based implementation is considered as one of the lesser challenging approaches in expert system implementation. For complicated systems, Fuzzy Inference Systems (FIS) is preferable over simple rule base systems. FIS is a widely used technique in various domains, and uses a fuzzy rule base, in combination with fuzzy membership functions to make decisions. FIS offers built-in expert knowledge and maps inputs to outputs employing the IF-THEN base rule. One such technique to detect drowsiness in drivers involved a fuzzy system, which Mouth and eye state as input to FIS and the FIS deduced the driver state as fit, fatigue or dangerous. The eyes state was categorized as blink, sleepy and slept, while mouth state was categorized as normal and yawning. In a more recent study, characteristics such as eye state and mouth state are fed to two layered FIS to deduce the level of fatigue of driver.

Advantages of using a FIS are:

- provides a high degree of flexibility and is useful in many vision based applications
- offers data less training and provides parallel processing as all the rules are applied simultaneously
- have the ability to learn by incorporating additional rules and knowledge base

3.1.3 MACHINE LEARNING BASED MODELS:

Machine learning based implementations are data driven algorithms trained on extensive driving data acquired in laboratory and on the road testing. The category can be broadly divided into shallow and deep models based on the levels of representation and the technique used for feature derivation.

3.1.3.1 SHALLOW MODELS

Shallow models provide reasonable predictive ability with minimal complexity. Shallow models consist of a few layers and require limited training data, but they require predefined discriminative features. Well known techniques in this domain include Artificial Neural Networks (ANN) with one hidden layer and Support Vector Machine (SVM).

In various techniques, ANNs and SVMs are trained on features like PERCLOS (% of eyelid closure), EEG and ECG signals of the driver, and other features, using in various permutations and combinations, to classify a driver as alert, drowsy or dangerous.

However, as mentioned earlier, these techniques use precomputed features. Due to inability to determine

these factors accurately, deep models are used.

3.1.3.2. DEEP MODELS

Deep learning models are machine learning techniques which incorporate learning representation of data instead of task specific methods. In contrast to shallow models, deep models have the ability to extract the features from the training data. Convolutional neural networks (CNN) based models are primarily used for driver fatigue detection.

Our project is based on this technique, to incorporate a deep learning based 3D convolutional neural networks to extract features from a video-based input of a driver's face to detect his/her state as active or drowsy.

3.2. FEATURES USED FOR DROWSINESS DETECTION

Various features incorporated in drowsiness detection are given in the following figure.

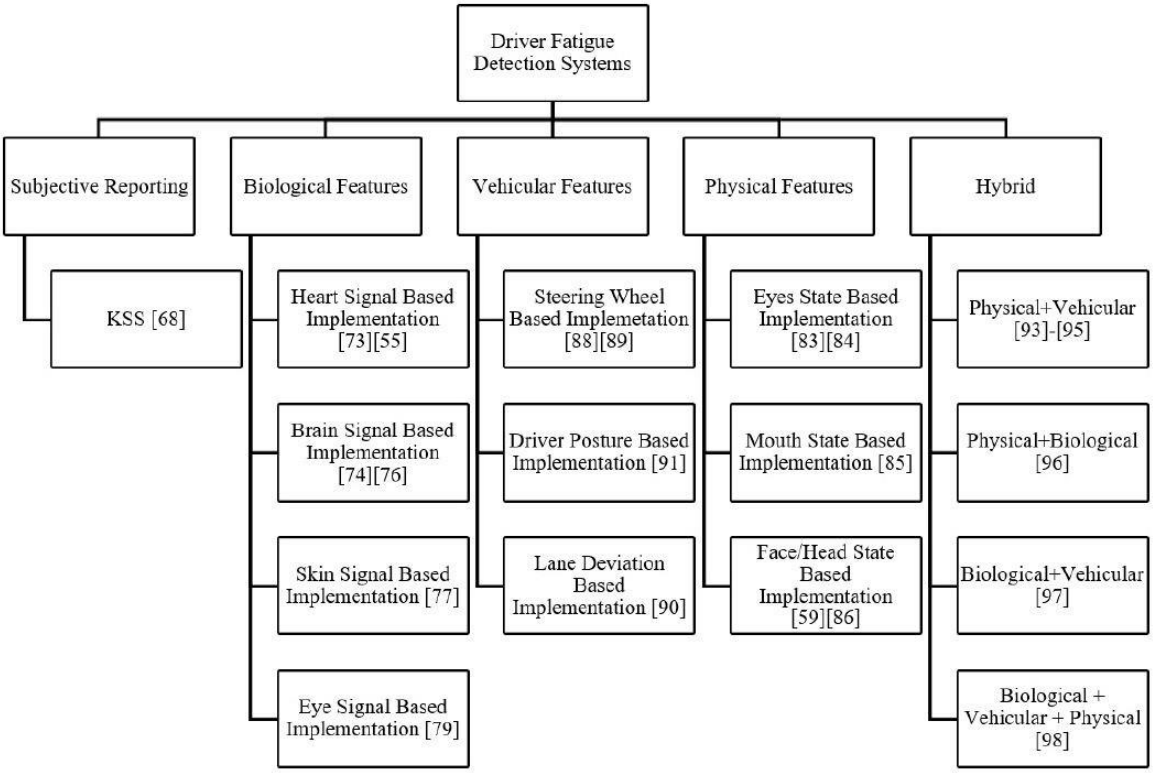


***Figure. 2**. Fatigue Detection Features*

These features are used based on various requirements, and possess certain traits, which are, in brief, given in Table I.

On looking at various parameters associated with features to be used, we see that the biological features are intrusive in nature, due to which the driving experience and other factors of the driver will get hampered. Further, it does not have real time applicability, and the installation costs are high.

Simultaneously, vehicular features are not that accurate, because it will depend a lot on the surrounding, eg. A turning road will have a lot of movement of the steering wheel, and it can still detect this as a drowsy driver. Simultaneously, installation cost is also high in some of the techniques, and hampers real-time applicability.

Whereas, in physical features, these disadvantages are not present. There is some dependency on the brightness of the surrounding, but still, other advantages make this way better as compared to using other features. So, in this project, physical features are used to determine the state of the driver.

***TABLE I**: COMPARISON OF VARIOUS FEATURES*

Category	**Signal**	**Parameter**	**Contact**	**Cost**	**Real-time Applicability**	**Limitations**
Biological Features	Brain	EEG	Yes	Low	No	Extremely Intrusive Prone to human movement
	Heart	ECG	Yes	High	No	
	Skin	sEMG	Yes	High	No	
Vehicular Features	Steering	SWA	Yes	High	Yes	Driver and Environment Dependency
	Lane	Lane Deviation	No	Low	Yes	
	Posture	Pressure	Yes	High	Yes	
Physical Features	Eyes	PERCLOS, Blink	No	Low	Yes	Illumination and Background Dependency
	Mouth	Yawn	No	Low	Yes	
	Face	Nod	No	Low	Yes	
	Nose	Structure	No	Low	Yes	

3.3 Python:

3.3.1 Introduction

Python is an interpreted , high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was created in the late 1980s, and first released in 1991, by **Guido van Rossum** as a successor to the ABC programming language. Python 2.0, released in 2000, introduced new features, such as list comprehensions, and a garbage collection system with reference counting, and was discontinued with version 2.7 in 2020. Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible and much Python 2 code does not run unmodified on Python 3. With Python 2's end-of-life, only Python 3.6.x and later are supported, with older versions still supporting e.g. Windows 7 .

3.3.2 Data Structures Used

3.3.2.1 Python List

The list is a most versatile data type available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example –

```
list1 = ['physics', 'chemistry', 1997, 2000]
list2 = [1, 2, 3, 4, 5 ]
list3 = ["a", "b", "c", "d"]
```

Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on.

3.3.2.2 Python Tuple

A tuple is a collection of objects which ordered and immutable. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example –

```
tup1 = ('physics', 'chemistry', 1997, 2000)
tup2 = (1, 2, 3, 4, 5)
tup3 = ("a", "b", "c", "d")
```

The empty tuple is written as two parentheses containing nothing –

```
tup1 = ();
```

To write a tuple containing a single value you have to include a comma, even though there is only one value –

```
tup1 = (50,);
```

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

3.3.2.3 Python Dictionary

Dictionary in Python is an unordered collection of data values, used to store data values like amap, which unlike other Data Types that hold only single value as an element, Dictionary holds **key: value** pair. Key value is provided in the dictionary to make it more optimized.
Note – Keys in a dictionary doesn't allows Polymorphism.

Creating a Dictionary-
In Python, a Dictionary can be created by placing sequence of elements within curly {} braces, separated by 'comma'. Dictionary holds a pair of values, one being the Key and the other corresponding pair element being its Key: value. Values in a dictionary can be of any data type andcan be duplicated, whereas keys can't be repeated and must be *immutable*.

Note – Dictionary keys are case sensitive, same name but different cases of Key will be treated distinctly.

```
# Creating a Dictionary#
with Integer Keys

Dict = {1: 'AISSMS', 2: 'IOIT', 3: 'PUNE'}
print("\n Dictionary with the use of Integer Keys: ")
print(Dict)

# Creating a Dictionary#
with Mixed keys
Dict = {'Name': 'IOIT', 1: [1, 2, 3, 4]}
print("\n Dictionary with the use of Mixed Keys: ")
print(Dict)
```

Output:

Dictionary with the use of Integer Keys:
{1: ‘AISSMS’, 2: ‘IOIT’, 3: ‘PUNE’}

Dictionary with the use of Mixed Keys:
{1: [1, 2, 3, 4], 'Name': ‘IOIT’}

Dictionary can also be created by the built-in function dict(). An empty dictionary can becreated by just placing to curly braces {}.

3.4 In-build Modules Used:

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarilynamed attributes that you can bind and reference.

Simply, a module is a file consisting of Python code. A module can define functions, classesand variables. A module can also include run able code.

Example

The Python code for a module named *aname* normally resides in a file named *aname.py*. Here'san example of a simple module, support.py

```
def print_func( par ):
print "Hello : ", parreturn
```

The *import* Statement
You can use any Python source file as a module by executing an import statement in some otherPython source file. The *import* has the following syntax –

```
import module1[, module2[,... moduleN]
```

When the interpreter encounters an import statement, it imports the module if the module is present in the search path. A search path is a list of directories that the interpreter searches before importing a module. For example, to import the module support.py, you need to put the followingcommand at the top of the script –

```
# Import module support
import support

# Now you can call defined function that module as follows
support.print_func("Zara")
```

When the above code is executed, it produces the following result –Hello

: Zara

A module is loaded only once, regardless of the number of times it is imported. This preventsthe module execution from happening over and over again if multiple imports occur.

3.4.1 OpenCV

- OpenCV (*Open Source Computer Vision Library*) is a library of programming functions mainly aimed at real-time computer vision.
- It was originally developed by Intel
- It will be used for facial and emotion recognition.

3.4.2 NumPy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

In our project we will keep track of the facial coordinates using the NumPy package as it contains powerful tools to operate on the coordinates.

3.4.3 Keras

Keras is an open-source software library that provides a Python interface for artificial neural networks.

Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

3.4.4 Imutils

Imutils are a series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV.

OpenCV can be a big, hard to navigate library, especially if you are just getting started learning image processing. So therefore, imulits helps you to work on images conveniently.

3.4.5 SciPy

SciPy is a free and open-source Python library used for scientific computing and technical computing.

SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

3.4.6 Math

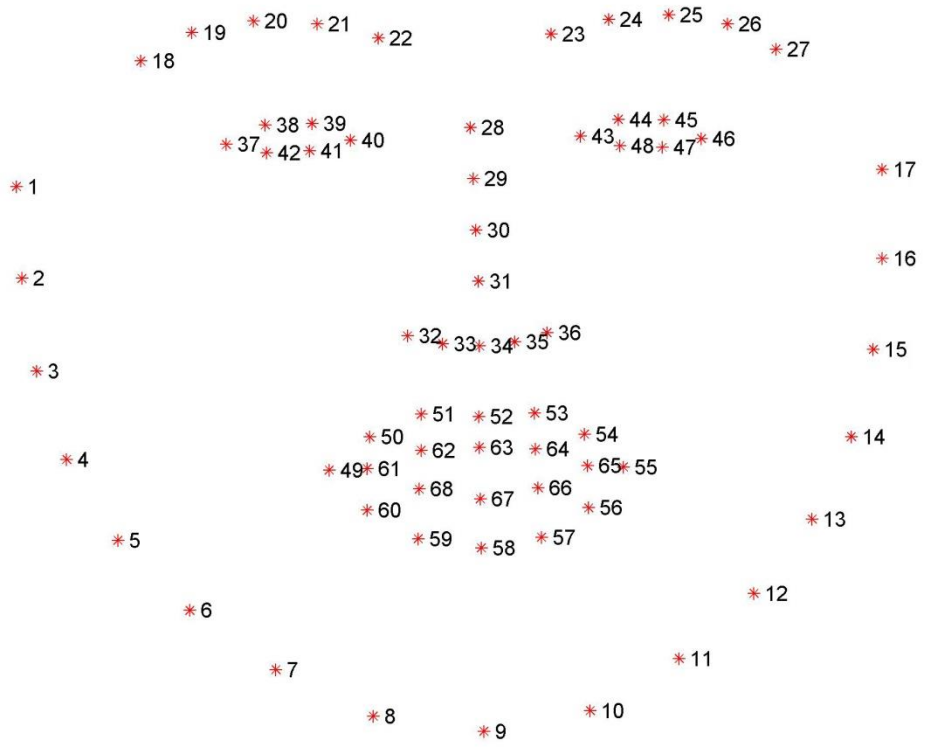
The math package is used to perform mathematical operations or tasks on the coordinate system provided by the dlib package.

3.4.7 Dlib

It‘s a landmark’s facial detector with pre-trained models, the dlib is used to estimate the location of 68

coordinates (x, y) that map the facial points on a person's face like image below.

These points are identified from the pre-trained model



5. ALGORITHM

5.1Driver Drowsiness Detection

1. START
2. Import OS, cv2, numpy as np, load_model from keras.model, and mixer from pygame
3. For face and eye detection using open cv as cv2.CascadeClassifier
4. Setting camera to initial state
5. Load Trained model
6. Convert the real time image in gray colour
7. Detection of eyes and face from real time images
8. Drawing rectangle frame for face and both eyes using cv2
9. Normalizing data from real time image to get reading between 0 and 1.(0 for closed eyes and 1 for open eyes)
10. Eye condition status as per result of real time image from trained model
11. Adding beep sound using mixer from pygame
12. Display and Break condition
13. STOP.

5.2Model Training

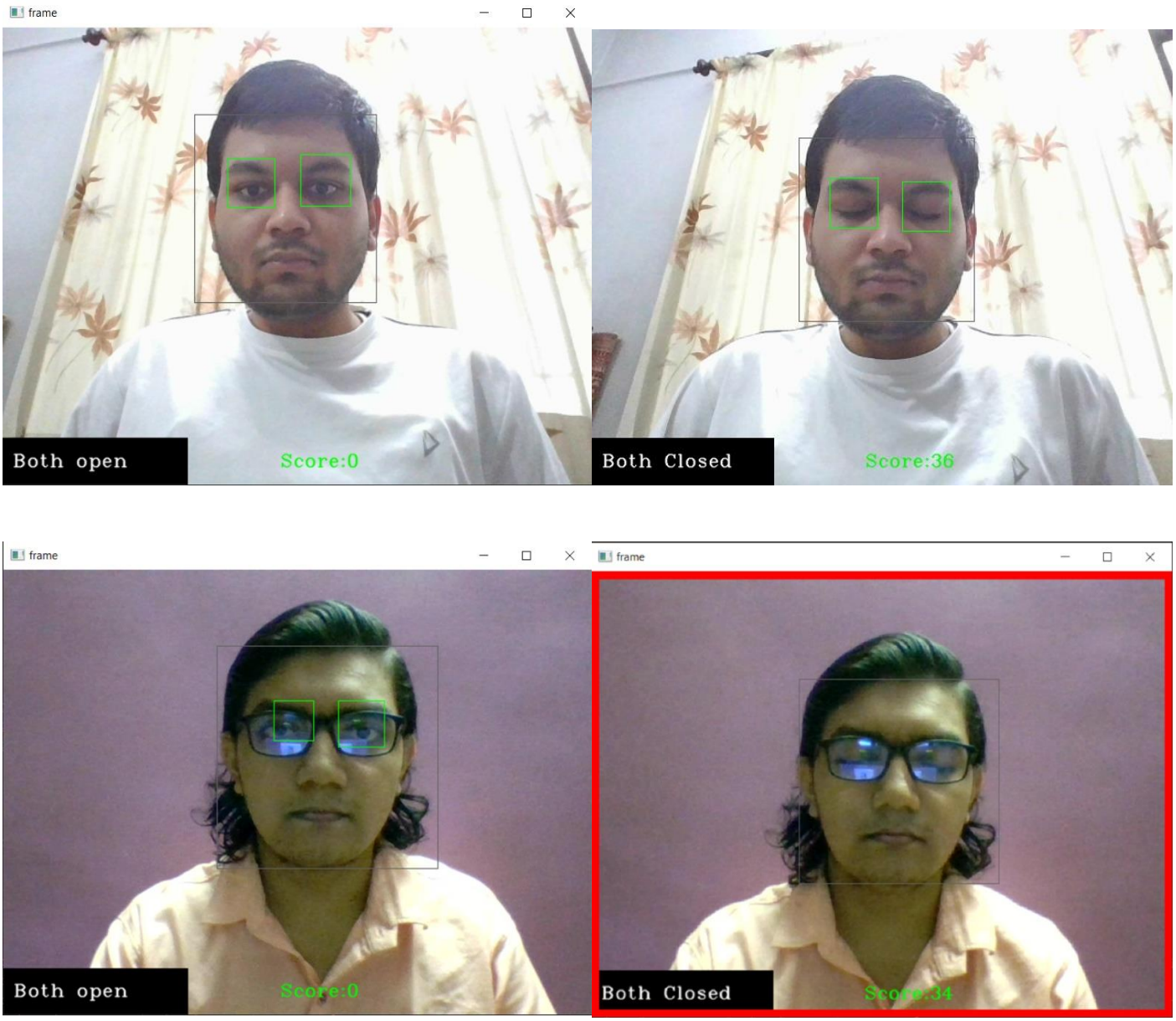
1. START
2. Import tensorflow as tf
3. Import cv2
4. Import matplotlib.pyplot as plt
5. Import numpy as np
6. Import os
7. Import keras module from tensorflow.
8. Import the data files of all the eyes (open and close eyes) and store the dataset.
9. Divide the dataset into two classes, one for open eye dataset and another for closed eye dataset.
10. Set standard image size as 224.
11. Covert each image in the dataset to RGB format.
12. Resize every image from the dataset to the standard image size (224)
13. Load the complete modified dataset into an array.
14. If any exception is encountered while loading, then pass the exception.
15. Create the training data and show the number of images present in it.
16. Randomly shuffle the training data before testing.
17. Reshape the complete training data and make the input size of each image in it same.
18. Normalize the training data (by dividing it by 225.0)
19. Save the trained data.
20. Load the saved training data.
21. Load the haar cascade file for creating different layers on the input model.
22. Create flat layers on the input model using the keras module.
23. Save the newly created module.

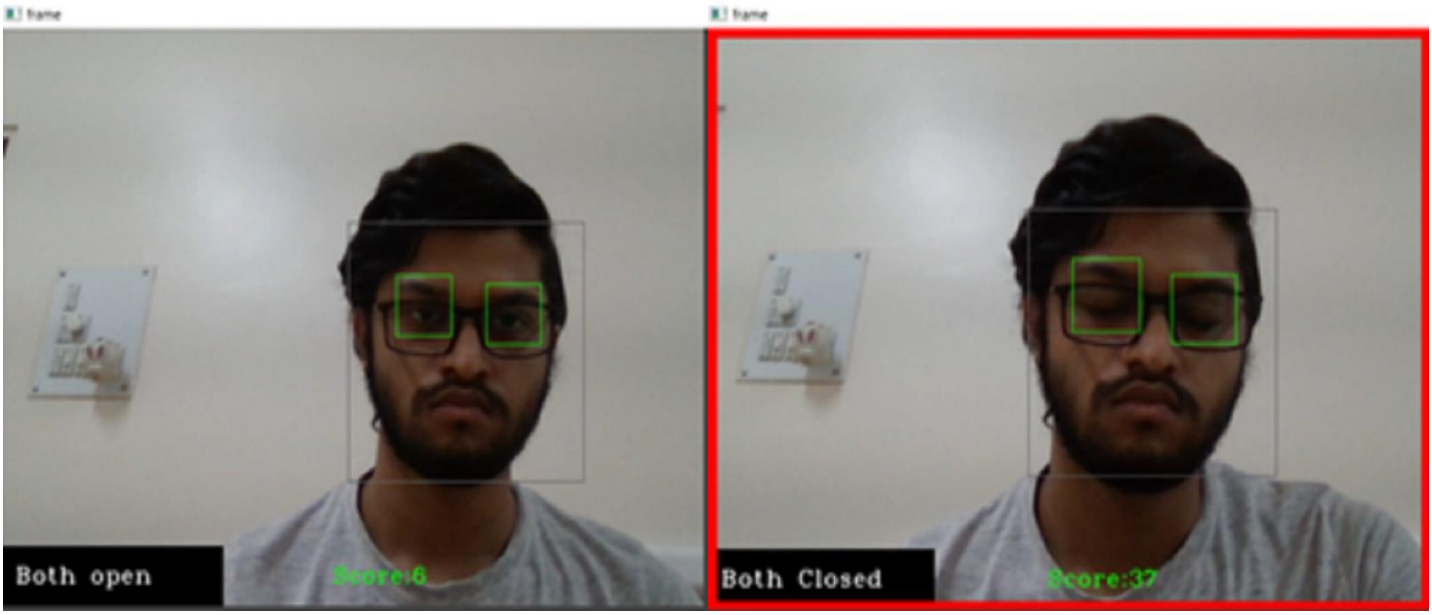
    // Checking network predictions of the loaded model.
24. From the dataset select one image from random and check whether the model is able to load the image.
25. Check the prediction of the model (eye open or closed) for that particular image.

    // Testing model on different faces.
26. Select random images which are not present in the dataset.
27. Load the images in a variable to proceed to further testing.
28. Convert the image to RGB system and display it.
29. Process the image in the haar cascade file to detect the frontal face of the person.
30. Now, process the image into our model to detect the eyes of the person.
31. Show the co-ordinates of the detected eyes of the image.
32. Split the co-ordinates into two parts i.e. one for left eye and another for right eye.
33. Now draw a rectangular box over the eye using the co-ordinates we obtained and display the image.
34. Now we have successfully detected the eyes of the person.
35. Crop the image such that we obtain the left and right eyes only so that we can proceed to further detection (i.e. whether the eye is open or closed).
36. Show the shape of each eye to the user.
37. Resize the cropped images of the eyes to our standard size of the prediction dataset (which is 224).
38. Now we have normalized the input images to the coordinates of our dataset.
39. Pass both eye images to the training model.

40. Display the predicted result of both the eyes separately.
41. The result is close to 1 of eye is open while the result is close to zero if eye is closed.
42. Link the model to the python file which will send the data of the live streaming to the particular model.
43. After receiving the data the model will predict the output and send the results to the live streaming.
44. STOP

6. SCREENSHOTS OF OUTPUT





ADVANTAGES OF THE SYSTEM

1. Security of vehicle and Driver.
2. Reliable and very fast.
3. Up to 90% Accuracy(Accuracy ca be increase using different models and Vehicle based Methods).
4. Non-Intrusive and Ease of Use.
5. Cost Efficiency is more than other systems.

8.DISADVANTAGES OF THE SYSTEM

1. Mis detects the eye corners as the eye centers.
2. Light Conditions and Background may affect the Detection of Drowsiness.

9.Future Enhancement

1. Vehicle Based Detection Features
2. In case of accident share location of vehicle to ambulance, police, fire brigade and 5 close peoples of driver.
3. Earlier detection usning Physiological Method.

10.**APPLICATIONS**

1. Best use of this system is in Trucks and Bus as they have long journey to travel.
2. Further we can use this systems in different modes of transport.

11.CONCLUSION

1. The process of drivers' drowsiness detection is very important for both individual and community safety
2. The growing use of Artificial Intelligence can be immensely useful in predicting fatigue of a driver
3. Our model automatically predicts a driver as drowsy or not by recognizing and key features from its face and predicting the output in real-time with an accuracy of 73.2%
4. It has various advantages as the installation is simple - of a camera, and it does not hinder the driver's experience and comfort like in other techniques such as biological features or vehicle-based features

12.REFRENCES

1. Schmidhuber J (2015) Deep learning in neural networks: an overview. Neural Netw 61:85–117. https://doi.org/10.1016/j.neu net.2014.09.003
2. Silberman N, Guadarrama S (2016) TensorFlow-Slim image classification model library. https://github.com/tensorflow/mo dels/tree/master/research/slim
3. Simonyan K, Zisserman A (2014) Two-stream convolutional networks for action recognition in videos. In: Ghahramani Z, Welling M, Cortes C, Lawrence ND, Weinberger KQ (eds)
4. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC (2018) MobileNetV2: inverted residuals and linear bottlenecks. 2018 IEEE conference on computer vision and pattern recognition (CVPR). IEEE, Salt Lake City, UT, pp 4510–4520
5. Advances in neural information processing systems 27. Curran Associates Inc, Montreal, pp 568–576. http://papers.nips.cc/paper/5353-two-stream-convolutional-networks-for-action-recognition-in-videos.pdf Soomro K, Zamir AR, Shah M (2012) UCF101: a dataset of 101 human actions classes from videos in the wild. Tech. Rep. CRCV-TR-12-01, Center for Research in Computer Vision, University of Central Florida, Orlando, FL
6. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: 2015 IEEE conference on computer vision and pattern recognition (CVPR), IEEE, Boston, MA, pp 1–9. https://doi.org/10.1109/CVPR.2015.7298594
7. Wijnands, J.S., Thompson, J., Nice, K.A. et al. Real-time monitoring of driver drowsiness on mobile platforms using 3D neural networks. Neural Comput & Applic 32, 9731–9743 (2020). https://doi.org/10.1007/s00521-019-04506-0