DROWSINESS DETECTION

PROJECT REPORT

18CSE484T – DEEP LEARNING

(2018 Regulation)

III Year/ VI Semester

Academic Year: 2022 -2023

By

SAHIL SATASIYA(RA2011026010110)
VARUN KHACHANE (RA2011026010072)
SHRESTH GUPTA (RA2011026010091)

Under the guidance of

Dr. Maivizhi R

Assistant Professor

Department of Computational Intelligence

FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
Kattankulathur, Kancheepuram
MAY 2023

SRM

INSTITUTE OF SCIENCE & TECHNOLOGY

SRM INSTITUTION OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR-603203

BONAFIDE CERTIFICATE

Certified that this Course Project Report titled **"DRIVER DROWSINESS DETECTION USING CNN"** is the bonafide work done by **VARUN KHACHANE [RA2011026010072], SHRESTH GUPTA [RA2011026010091], and SAHIL SATASIYA [RA2011026010110]** who carried out under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

R. Maivizhi 17/5/22

SIGNATURE
Faculty In-Charge
Dr.MAIVIZHI R
Assistant Professor
Department of Computational Intelligence
SRM Institute of Science and Technology
Kattankulathur Campus, Chennai

HEAD OF THE DEPARTMENT
Dr. R Annie Uthra
Professor and Head ,
Department of Computational Intelligence,
SRM Institute of Science and Technology
Kattankulathur Campus, Chennai

TABLE OF CONTENT

ABSTRACT

Drowsiness is a state of near sleep, where the person has a strong desire for sleep. It has two distinct meanings, referring both to the usual state preceding falling asleep and the chronic condition referring to being in that state independent of a daily rhythm. Sleepiness can be dangerous when performing tasks that require constant concentration, such as driving a vehicle. When a person is sufficiently fatigue while driving, they will experience drowsiness and this leads to increase the factor of road accident.

The development of technologies for detecting or preventing drowsiness while driving is a major challenge in the field of accident-avoidance system. Because of the hazard that drowsiness presents on the road, methods need to be developed for counteracting its affects.

The main idea behind this project is to develop a nonintrusive system which can detect fatigue of any human and can issue a timely warning. Drivers who do not take regular breaks when driving long distances run a high risk of becoming drowsy a state which they often fail to recognize early enough.

This system will monitor the driver eyes using a camera and by developing an algorithm we can detect symptoms of driver fatigue early enough to avoid the person from sleeping. So, this project will be helpful in detecting driver fatigue in advance and will give warning output in form of alarm and popups to prevent any type of harm either to them or to the surroundings.

MODEL DESCRIPTION

CNN (Convolutional Neural Network):

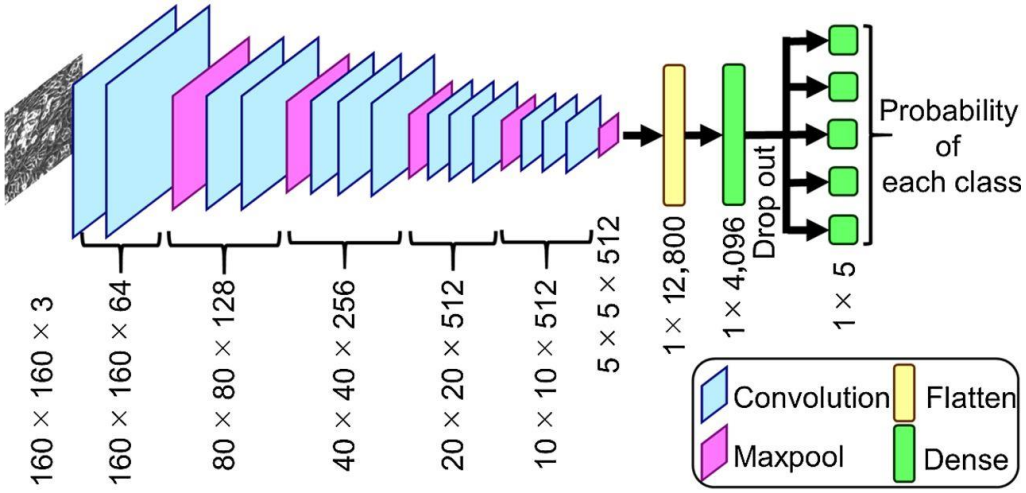
Convolutional Neural Networks (CNNs) are a class of deep learning models specifically designed for processing and analyzing visual data such as images and videos. CNNs have revolutionized computer vision tasks by achieving state-of-the-art performance in various domains, including object recognition, image classification, and, relevant to this project, drowsiness detection.

The key components of a CNN are convolutional layers, pooling layers, and fully connected layers. Here is an overview of each component:

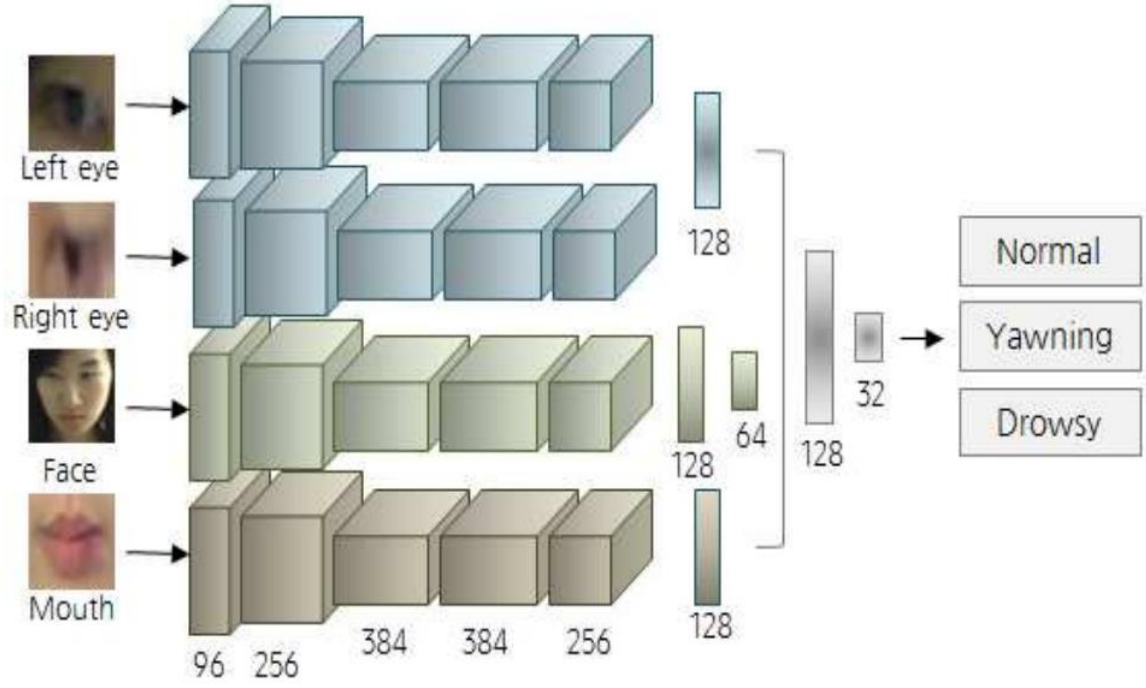
1. Convolutional Layers: Convolutional layers are responsible for extracting features from input data. These layers consist of filters or kernels, which are small matrices that slide or convolve over the input image, performing element-wise multiplication and summation operations. This process helps to capture important visual patterns and spatial relationships in the data. Multiple convolutional layers can be stacked to learn increasingly complex and abstract features.
2. Pooling Layers: Pooling layers are inserted between convolutional layers to reduce the spatial dimensions of the feature maps. The most common pooling operation is max pooling, which down samples the feature maps by selecting the maximum value within each pooling window. This helps to reduce computational complexity and make the model more robust to variations in the input data.
3. Fully Connected Layers: Fully connected layers, also known as dense layers, are responsible for performing classification or regression tasks based on the extracted features. These layers connect every neuron from the previous layer to every neuron in the current layer. The output of the fully connected layers is typically fed into a softmax activation function to generate probability distributions over different classes.

DIAGRAMS

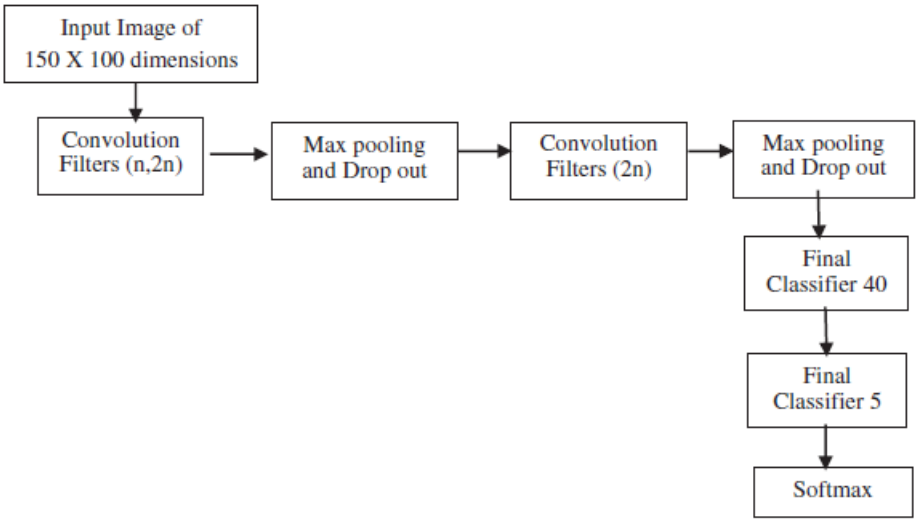
CNN Architecture:

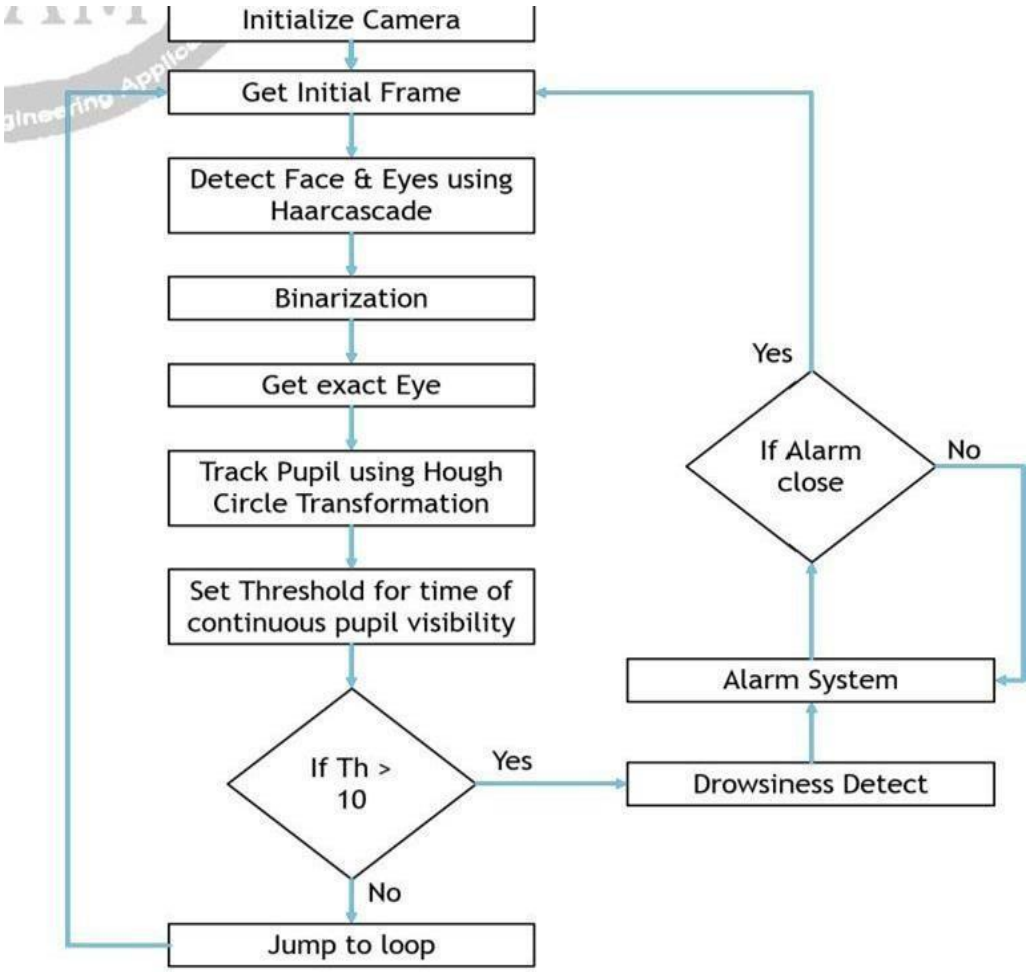


System Architecture:

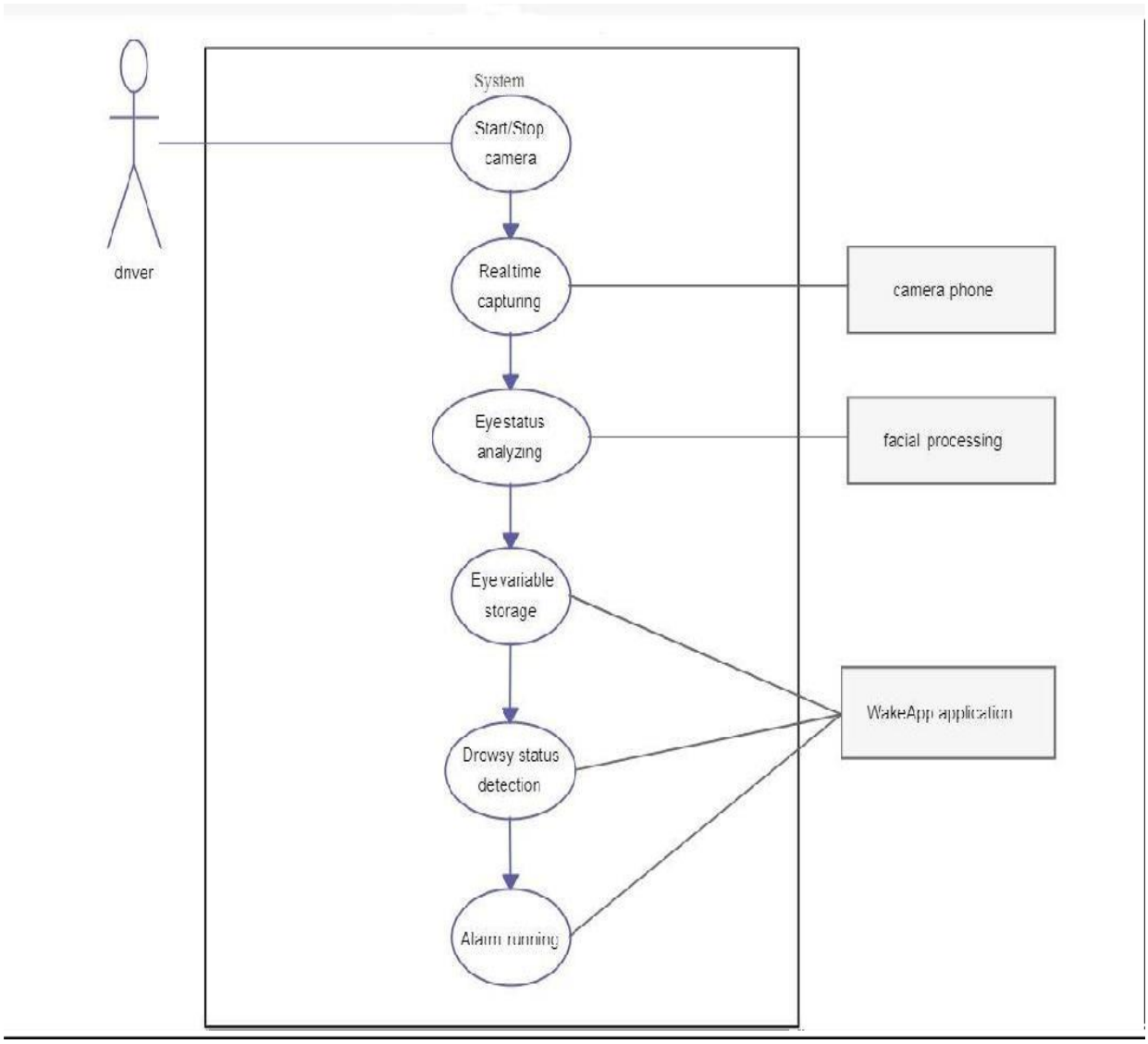


Module Architecture:





Use Case Diagram:



LITERATURE SURVEY

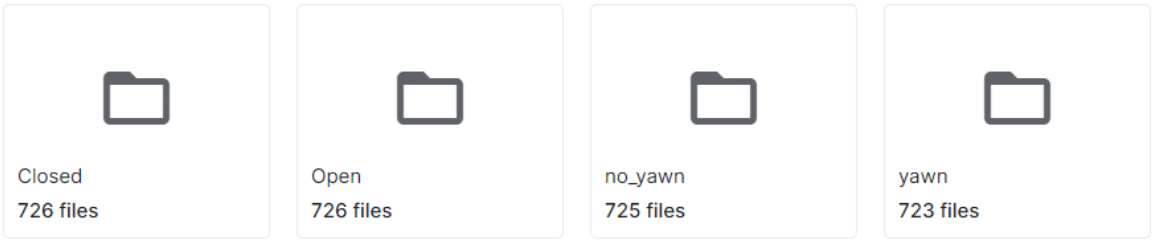
Name of the Paper	**Authors**	**Published in**	**URL**
DRIVER DROWSINESS DETECTION SYSTEM	Belal ALSHAQAQI; Abdullah Salem BAQUHAIZEL; Mohamed El Amine OUIS; Meriem BOUMEHED; Abdelaziz OUAMRI; Mokhtar KECHE	IEEE	https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6602353
Real-Time Driver Drowsiness Detection using Computer Vision	Mahek Jain, Bhavya Bhagerathi, Sowmyarani CN	International Journal of Engineering and AdvancedTechnology (IJEAT) ISSN: 2249-8958 (Online),Volume-11 Issue-1, October 2021	https://www.ijeat.org/wp-content/uploads/papers/v11i1/A31591011121.pdf
Driver Drowsiness Detection by Applying Deep Learning Techniques to Sequences of Images	Elena Magán , M. Paz Sesmero , Juan Manuel Alonso-Weberand Araceli Sanchis	MDPI	https://www.mdpi.com/2076-3417/12/3/1145

DATASET DESCRIPTION

https://www.kaggle.com/datasets/dheerajperumandla/drowsiness-dataset

The dataset contains 4 folders namely:

1. Closed: 726 files
2. Open: 726 files
3. No_yawn: 725 files
4. Yawn: 723 files



Closed:

Open:

No_yawn:

Yawn:

MODULES AND ANALYSIS

The modules the proposed system contain are:

- Data pre-processing
- Feature Extraction
- Training the system
- Testing the system

DATA PRE-PROCESSING

The images are tested with the proposed algorithm, preprocessing is accomplished which is the primary stage of any face recognition gadget. A brand-new approach of preprocessing has been proposed for face recognition packages beneath uncontrolled and difficult lighting fixtures situations.

FEATURE EXTRACTION

As in brief alluded to in advance, primarily based at the facial landmarks that we extracted from the frames of the videos, we ventured into growing appropriate capabilities for our class version. At the same time as we hypothesized and examined several functions, the four core features that we concluded on for our very last models had been eye element ratio, mouth thing ratio, student circularity, and sooner or later, mouth element ratio over eye thing ratio. Eye Aspect Ratio (EAR) EAR, is the ratio of the lengthand the width of the eyes. The length and width of the eyes is calculated by means of averaging over the two horizontal and vertical lines throughout the eyes as illustrated. Our speculation turned into that once an man or woman is drowsy, their eyes are probable to get smaller and they are probable to blink extra. Based totally On this speculation, we anticipated our model to expect the class as drowsy if the EAR for an character over successive frames declined i.e. Their eyes began to be closeor they were blinking faster.

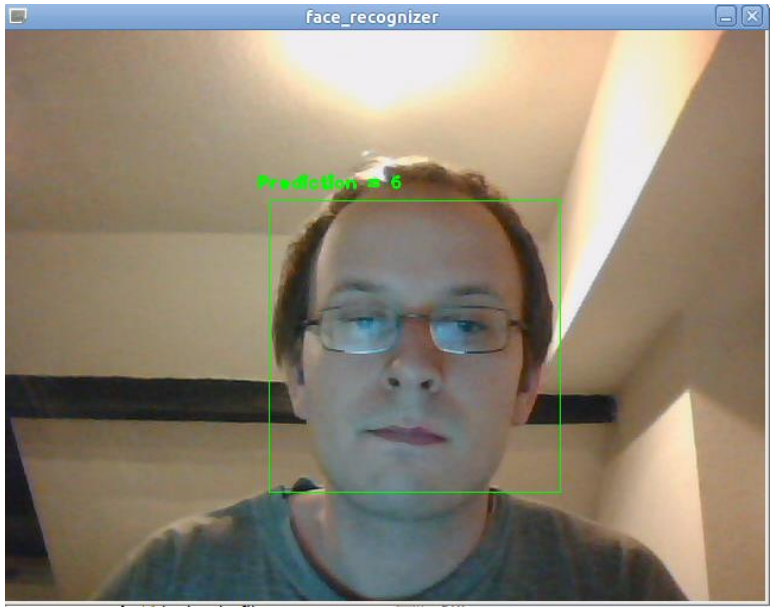
TRAINING THE SYSTEM

To do face recognition, face recognizer have to learn. The use of the pre-categorized dataset, we have designed a label “dataset” for our face recognition system, which is now used to use that dataset, which first trains the face identifier to use the dataset in OpenCV Python. Python Trainer in the same folder.p "File, and then create a folder in the same directory" Trainer. ", This is the binder where we will save our identifier after coaching. For our training and test data for drowsiness detection, we used the (shape_predictor_68_face_landmarks.dat) . Which is a dataset trained based on the ibug 300-w dataset.

METHODOLOGY

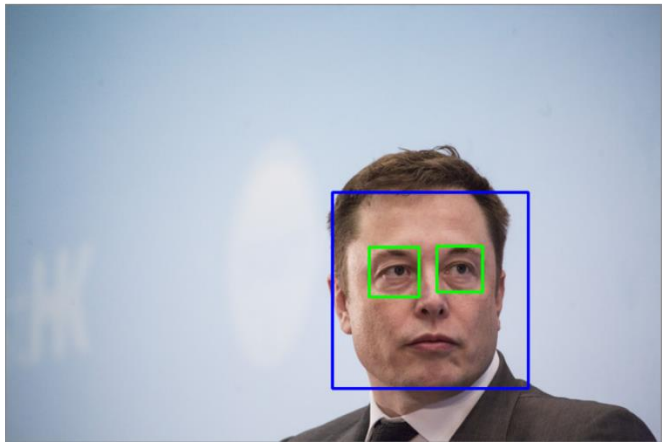
Face Detection

The suggested system's face detection is based on the Viola-Jones object detection framework. Because all human faces have comparable qualities, the Haar-features can be used. Despite its age, this framework, like with many of its CNN competitors, is still the best in face detection. In order to construct a system that detects things quickly and accurately, this framework incorporates the principles of Haar-like Features, Integral Images, the AdaBoost Algorithm, and the Cascade Classifier. Haar-Features can be used to determine features that are similar to the eye region darker than upper and those that seem similar to the nose bridge region brighter than eyes. These are computed not from the original image, however from the integral image. At any location(x,y), the integral image is the sum of pixels above and to the left of (x,y).



Eyes Detection

The open eye detection cascade classifiers in OpenCV work in a similar way. Classifiers are available in the OpenCV library for recognising different eye orientations such as open, closed eyes with eyeglasses but no sunglasses. Haarcascade eye.xml, which identifies open eyes, is the fastest of all. If the user is wearing glasses, the classifier haarcascade eye tree eyeglasses.xml is picked according to his preferences. To avoid false positives from the backdrop, eyes are only identified in the part of the face that is visible. The minimum size of the eyes is 20*20. The face detected area displays open eyes that have been detected. A timer is begun as soon as no eyeballs are detected. When your eyes are closed, the system fails to detect the eyeballs and thus triggers the alarm. If the driver's eyes are closed for more than the counter value, the system determines that the driver is tired and sounds an alarm. All of this is done through a python application interface.



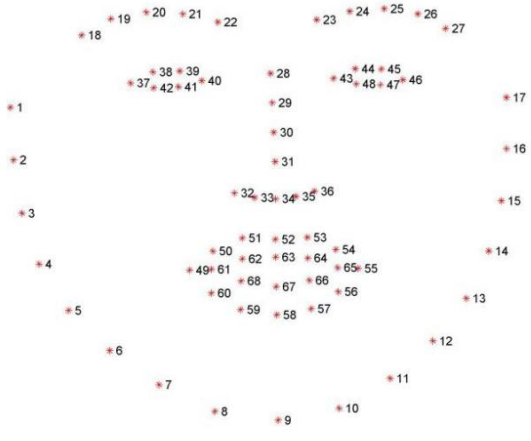
LIBRARIES

HAAR CASCADE XML

Haar cascade xml is an method primarily based on machine getting to know(ml) in which lots of tremendous and negative images are used to teach the classifier. Superb pictures are the ones pictures that incorporate the photographs which we need our classifier to become aware of as an example: a face and negative pix are the images of the whole thing else i.E they are the pictures which do now not contain the item we need to discover. This helps us in face detection so that the device can efficaciously come across a face so that it will method the face reputation method. Haar cascade classifier is a very effective object detection technique which became popular via paul viola and michael jones via their paper,“speedy object detection the use of a boosted cascade of simple functions” in 2001. ... Based on the training it miles can be used to identify certain objects in different pictures The Haar Cascade classification is based on Harvowlet technology to analyze the pixels in an image In sections by function. It uses the concept of "comprehensive image" to calculate "features"Need to know. Haar cascade makes use of the the ada-increase mastering set of rules to pick a small range from a massive set of statistics to provide an powerful result of essential capabilities from a big set and cascading method is used to identify the face within the picture.

D-LIB

D-lib is a toolkit used for making realtime machine learning applications and information evaluation packages. It’s used for face detection/ face recognition and facial landmark detection. The frontal face detector in d-lib works really nicely as it's miles quite simple and it simply works out of the box.
Dlib has a 68 points model. Right here we will see that it factors from 1 to 68. However on occasion we don't want all the 68 feature points, consequently, we can personalize those factors consistent with our requirements as an example in our machine we simplest use the factors 37 to forty eight for calculating the EAR



OpenCV

OpenCV (Open Source Computer Vision Library) is a free software library for computer vision and machine learning. Apart from the necessity to speed the usage of machine viewpoints in commercial goods, it was created to provide a standard architecture for computer vision applications. This library includes a comprehensive set of both classic and state-of-the-art computer vision, as well as machine learning algorithms, with over 2500 optimised algorithms. To detect and recognize faces, identify objects, classify human actions in videos, track moving objects, track camera movements, produce 3D point clouds from stereo cameras, extract 3D models of objects, combine images together producing a high resolution image of an entire scene, and similar images from an image database, remove red eyes from images taken using ash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. Over 500 algorithms and about 10 times as many functions that support those algorithms are included here. By including it as a library of image processing functions, it is easily deployed.

CODE SNIPPET

Face Detection to check whether a person is yawning or not:

```
def face_for_yawn(direc=r"./train", face_cas_path=r"data/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()
```

Eye Detection for Open or Closed Eye:

```
def get_data(dir_path=r"./train", face_cas=r"data/haarcascade_frontalface_default.xml", eye_cas=r"Required_files/haarcascade.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
```

Train Test Split:

```
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)
```

Importing the dependencies:

```
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
```

Data Augmentation:

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


#train_generator = tf.data.Dataset.from_tensor_slices((X_train, y_train))
#test_generator = tf.data.Dataset.from_tensor_slices((X_test, y_test))

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)
```

Model:

```
model = Sequential()

model.add(Conv2D(256, (3, 3), activation="relu", input_shape=(145,145,3)))
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 Execution:

```
history = model.fit(train_generator, epochs=50, validation_data=test_generator, shuffle=True, validation_steps=len(test_generator))
```

```
Epoch 1/50
40/40 [==============================] - 70s 2s/step - loss: 1.2364 - accuracy: 0.4344 - val_loss: 0.9665 - val_accuracy: 0.5075
Epoch 2/50
40/40 [==============================] - 69s 2s/step - loss: 0.6057 - accuracy: 0.7432 - val_loss: 0.4059 - val_accuracy: 0.8022
Epoch 3/50
40/40 [==============================] - 92s 2s/step - loss: 0.4226 - accuracy: 0.8192 - val_loss: 0.3172 - val_accuracy: 0.8993
Epoch 4/50
40/40 [==============================] - 112s 3s/step - loss: 0.3436 - accuracy: 0.8664 - val_loss: 0.2579 - val_accuracy: 0.8974
Epoch 5/50
40/40 [==============================] - 110s 3s/step - loss: 0.3165 - accuracy: 0.8720 - val_loss: 0.2271 - val_accuracy: 0.9123
Epoch 6/50
40/40 [==============================] - 100s 3s/step - loss: 0.2650 - accuracy: 0.8880 - val_loss: 0.2291 - val_accuracy: 0.9160
Epoch 7/50
40/40 [==============================] - 99s 2s/step - loss: 0.2475 - accuracy: 0.9000 - val_loss: 0.2957 - val_accuracy: 0.8769
Epoch 8/50
40/40 [==============================] - 100s 3s/step - loss: 0.2400 - accuracy: 0.9088 - val_loss: 0.1831 - val_accuracy: 0.9272
Epoch 9/50
40/40 [==============================] - 104s 3s/step - loss: 0.2273 - accuracy: 0.9064 - val_loss: 0.1930 - val_accuracy: 0.9235
Epoch 10/50
40/40 [==============================] - 105s 3s/step - loss: 0.2067 - accuracy: 0.9136 - val_loss: 0.1860 - val_accuracy: 0.9179
Epoch 11/50
40/40 [==============================] - 105s 3s/step - loss: 0.2055 - accuracy: 0.9144 - val_loss: 0.1566 - val_accuracy: 0.9310
Epoch 12/50
40/40 [==============================] - 100s 3s/step - loss: 0.1850 - accuracy: 0.9200 - val_loss: 0.2466 - val_accuracy: 0.8974
Epoch 13/50
40/40 [==============================] - 86s 2s/step - loss: 0.2227 - accuracy: 0.9072 - val_loss: 0.1900 - val_accuracy: 0.9254
Epoch 14/50
40/40 [==============================] - 84s 2s/step - loss: 0.2155 - accuracy: 0.9128 - val_loss: 0.1781 - val_accuracy: 0.9328
Epoch 15/50
40/40 [==============================] - 85s 2s/step - loss: 0.1772 - accuracy: 0.9344 - val_loss: 0.1420 - val_accuracy: 0.9534
Epoch 16/50
40/40 [==============================] - 87s 2s/step - loss: 0.1628 - accuracy: 0.9384 - val_loss: 0.1259 - val_accuracy: 0.9422
Epoch 17/50
40/40 [==============================] - 85s 2s/step - loss: 0.1652 - accuracy: 0.9312 - val_loss: 0.1383 - val_accuracy: 0.9496
Epoch 18/50
40/40 [==============================] - 85s 2s/step - loss: 0.1619 - accuracy: 0.9320 - val_loss: 0.1187 - val_accuracy: 0.9459
Epoch 19/50
40/40 [==============================] - 83s 2s/step - loss: 0.1404 - accuracy: 0.9480 - val_loss: 0.1280 - val_accuracy: 0.9496
```

Model Evaluation:

```
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, "b", label="training accuracy")
plt.plot(epochs, val_accuracy, "r", label="validation accuracy")
plt.legend()
plt.show()

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

Prediction:

0-yawn, 1-no_yawn, 2-Closed, 3-Open

```
# prepare("../input/drowsiness-dataset/train/no_yawn/1068.jpg")
prediction = model.predict([prepare(r"./train/no_yawn/1067.jpg")])
np.argmax(prediction)
```

```
1/1 [==============================] - 0s 97ms/step
1
```

```
prediction = model.predict([prepare(r"./train/Closed/_387.jpg")])
np.argmax(prediction)
```

```
1/1 [==============================] - 0s 29ms/step
2
```

```
prediction = model.predict([prepare(r"./train/Closed/_224.jpg")])
np.argmax(prediction)
```

```
1/1 [==============================] - 0s 29ms/step
2
```

```
prediction = model.predict([prepare(r"./train/yawn/12.jpg")])
np.argmax(prediction)
```

```
1/1 [==============================] - 0s 27ms/step
0
```

FINAL EXECUTION

Facial Landmarks for Eyes and Mouth:

We created start and end indices for both eyes and mouth

```
capture = cv2.VideoCapture(0)
avgEAR = 0
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
(leStart, leEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(reStart, reEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
(mStart, mEnd) = face_utils.FACIAL_LANDMARKS_IDXS["mouth"]
```

Checking whether a person is yawning or not:

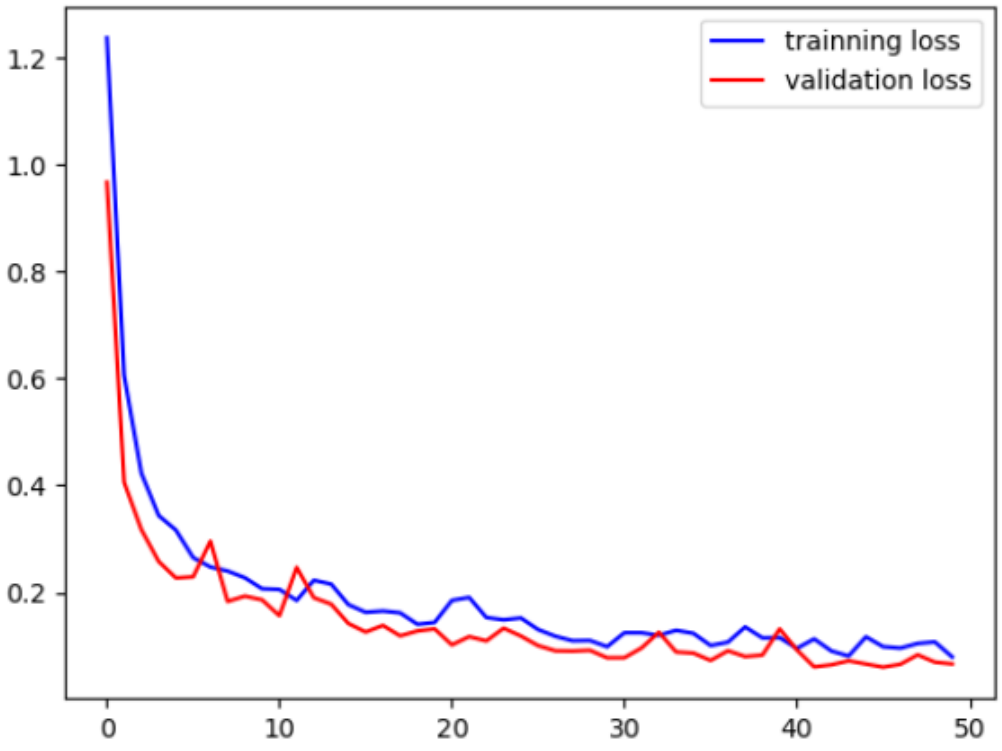
If the person is yawning, then the program will send an alert which will awaken the driver

```
if(yawn(shape[mStart:mEnd])>0.6):
    cv2.putText(gray, "Yawn Detected", (50,50), cv2.FONT_HERSHEY_COMPLEX, 1,(0,255,127),2)
    yawn_countdown=1

if(avgEAR<close_thresh):
    flag+=1
    eyeContourColor = (0,255,255)
    print(flag)
    if(yawn_countdown and flag>=frame_thresh_3):
        eyeContourColor = (147, 20, 255)
        cv2.putText(gray, "Drowsy after yawn", (50,50), cv2.FONT_HERSHEY_COMPLEX, 1,(0,255,127),2)
        alert.play()
        if(map_flag):
            map_flag = 0
            map_counter+=1
    elif(flag>=frame_thresh_2 and getFaceDirection(shape, size)<0):
        eyeContourColor = (255, 0, 0)
        cv2.putText(gray, "Drowsy (Body Posture)", (50,50), cv2.FONT_HERSHEY_COMPLEX, 1,(0,255,127),2)
        alert.play()
        if(map_flag):
            map_flag = 0
            map_counter+=1
    elif(flag>=frame_thresh_1):
        eyeContourColor = (0, 0, 255)
        cv2.putText(gray, "Drowsy (Normal)", (50,50), cv2.FONT_HERSHEY_COMPLEX, 1,(0,255,127),2)
        alert.play()
        if(map_flag):
            map_flag = 0
            map_counter+=1
elif(avgEAR>close_thresh and flag):
    print("Flag reseted to 0")
    alert.stop()
    yawn_countdown=0
    map_flag=1
    flag=0

if(map_counter>=3):
    map_flag=1
    map_counter=0
    vlc.MediaPlayer('take_a_break.mp3').play()
    webbrowser.open("https://www.google.com/maps/search/hospitals+near+me")
```

RESULTS AND DISCUSSION

Accuracy = 0.95

With an accuracy of 0.95, this model performs about as well as a randomguess.



OUTPUT

Drowsy Detection:

Normal:

Yawn Detection:

CONCLUSION

In conclusion, the creation of a sleepiness detection system is crucial to ensuring traffic safety and averting accidents brought on by fatigued drivers. A drowsiness detection system can efficiently monitor driving behaviour and send prompt notifications when indicators of tiredness are found by utilising cutting-edge technology like computer vision, machine learning, and signal processing.

The goal of the research was to develop a machine learning-based sleepiness detection system that uses features derived from facial landmarks and eye movements. A substantial dataset with a variety of driver samples and their related sleepiness labels was added into the system. The system proved its capacity to precisely classify driver tiredness based on the input features by training and evaluating machine learning models.

The architecture of the system integrated feature extraction algorithms, classification models, and image processing methods. The detection and tracking of face landmarks and eye movements were performed on real-time video input from a camera. The trained machine learning model was then fed the extracted features, and it generated predictions regarding the level of drowsiness of the driver.

The study of the sleepiness detection system demonstrated how well it can detect and warn of instances of drowsy driving. The system obtained great accuracy in differentiating between awake and asleep states by examining facial expressions, eye closure duration, and other pertinent aspects. Drivers may be prompted to take required precautions, such as taking a break, switching drivers, or using alertness-enhancing tactics, by the system's timely signals.

This project's sleepiness detection system offers a lot of promise for real-world uses in the auto industry, transportation businesses, and even personal vehicles. With constant monitoring of driver attentiveness and a decreased likelihood of accidents from drowsy driving, its inclusion into vehicles can act as an extra safety measure.

Although the established technology shows promising outcomes, more investigation and advancements can be made. By including further input modalities including steering wheel movements, vehicle speed, and physiological data, the system's performance can be improved. Additionally, diverse driving circumstances, driver appearances, and lighting conditions can be used to assess the system's robustness. This project can be used by automobile industries to prevent road accidents and thus prevent less deaths / injuries per year due to this.

REFERENCES

[1] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen. Compressing neural networks with the hashing trick. CoRR, abs/1504.04788, 2015. 2

[2] F. Chollet. Xception: Deep learning with depthwise separable convolutions. arXiv preprint arXiv:1610.02357v2, 2016. 1

[3] M. Courbariaux, J.-P. David, and Y. Bengio. Training deep neural networks with low precision multiplications. arXiv preprint arXiv:1412.7024, 2014. 2

[4] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. CoRR, abs/1510.00149, 2, 2015. 2

[5] J. Hays and A. Efros. IM2GPS: estimating geographic information from a single image. In Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 2008. 7

[6] J. Hays and A. Efros. Large-Scale Image Geolocalization. In J. Choi and G. Friedland, editors, Multimodal Location Estimation of Videos and Images. Springer, 2014. 6, 7

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, 2015. 1

[8] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015. 2, 7

[9] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. arXiv preprint arXiv:1611.10012, 2016.

[10] Real Time Drowsiness Detection using Eye Blink Monitoring by Amna Rahman. https://ieeexplore.ieee.org/document/7396336