DRIVER DROWSINESS DETECTION SYSTEM

A PROJECT REPORT

Submitted by

DHARANI R	**(19104013)**
KAVENESH N	**(19104036)**
NANDHAKUMAR S	**(19104044)**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

ERODE SENGUNTHAR ENGINEERING COLLEGE

(An Autonomous Institution)

PERUNDURAI, ERODE – 638 057.

JUNE 2022

BONAFIDE CERTIFICATE

Certified that this project report "**DRIVER DROWSINESS DETECTION SYSTEM**" is the bonafide work of **DHARANI R (19104013), KAVENESH N (19104036) and NANDHAKUMAR S (19104044),** who carried out the project work under my supervision. Certified further that to the best of my knowledge the reported herein does not from part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Mrs.S.SIVASELVI,	Dr.G.SIVAKUMAR,
SUPERVISOR	**HEAD OF THE DEPARTMENT**
Assistant Professor,	Professor and Head,
Department of Computer Science and Engineering,	Department of Computer Science and Engineering,
Erode Sengunthar Engineering College,	Erode Sengunthar Engineering College,
Erode-638 057.	Erode-638 057.

Submitted for End Semester Mini Project Viva -Voce Examination held on_______.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

First of all we would like to convey our heartfelt thanks to our respected Founder, **“UDYOG RATTAN” Deivathiru J.SUDHANANDHEN,B.Sc.,** who always blessed us to give the best.

We are delighted to thank our esteemed President **Thiru.V.ANNADURAI.,** & Correspondent **Thiru.G.KAMALAMURUGAN,BBM.,** of Erode Sengunthar Educational Trust and gave us full support for carrying out our project in adequate time and providing us with facilities.

We have immense pleasure to thank our Secretary **Thiru. S.N.THANGARAJU,B.E.,M.B.A..,** of Erode Sengunthar Educational Trust who gave us full support and guide in all the endeavours of our project.

We would like to express our thanks to our Principal **Dr.V.VENKATACHALAM,B.E.,M.S.,M.Tech.,Ph.D.,** for forwarding us to do our project and offering adequate duration in completing our project.

We render our sincere thanks to, **Dr.G.SIVAKUMAR,M.E.,Ph.D.,** Head, Department of Computer Science and Engineering for his anchoring support in doing this project.

We are grateful to our project coordinator **Ms.T.KALAISELVI,M.E.,(Ph.D)., Assistant Professor(Sl.G-I)** for her excellent ideas and encouragement.

We are heartly extend our thanks to our project guide **Ms.S.SIVASELVI,M.E., Assistant Professor** for her support, constant supervision and as well as for providing necessary information regarding the project.

ABSTRACT

Several studies have reported indicating that driver drowsiness has caused multiple number of accidents over the past years. So, in an effort to prevent such accidents, measures are to be implemented. One such measure is a Driver Drowsiness Detection software; the software will set off an alarm once the driver keeps his/her eyes closed for a certain amount of time. This paper proposes to detect whether the driver is sleepy or not by using OpenCV with eyelid related parameters. The data consists of around 1472 images of eyes under different optical conditions. The used dataset is divided into three parts for the sake of convenience i.e. frontal face detection, left eye detection and right eye detection. With all the features, an OpenCV and keras a driver's fatigue or drowsiness model is built using Convolution Neural Network (CNN). The validation results indicate the precision and accuracy of the proposed model.

Keywords : Drowsiness Detection, Image Processing, OpenCV, Convolution Neural Network.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO
	ABSTRACT	i
	LIST OF FIGURES	iv
	LIST OF ABBREVIATIONS	v
1	**INTRODUCTION**	1
2	**LITERATURE REVIEW**	2
	2.1 Drowsiness Detection System Utilizing Physiological Signals	2
	2.2 Drowsiness Detection with OpenCV (Using Eye Aspect Ratio)	5
	2.3 Real Time Driver Fatigue Detection Based on SVM Algorithm	6
	2.4 Deep CNN: A Machine Learning Approach for Driver Drowsiness Detection Based on Eye State	8
3	**SYSTEM ANALYSIS**	3
	3.1 Existing System	10
	3.1.1 Disadvantages	10
	3.2 Proposed System	10
	3.2.1 Advantages	10
4	**SYSTEM REQUIREMENTS**	11
	4.1 Hardware Requirements	11
	4.2 Software Requirements	11
5	**DESIGN OF THE SYSTEM**	12

6	**PROPOSED METHODOLOGY**	13
	6.1 Convolution Neural Networking	14
	6.1.1 Convolution Layer	15
	6.1.1.1 Pooling Layer	18
	6.1.1.2 Max pooling	18
	6.1.1.3 Fully Connected Layer	19
7	**SYSTEM DESIGN**	20
	7.1 Systametic flow diagram	20
8	**SYSTEEM MODULES**	22
	8.1 Collecting the Dataset	22
	8.2 Face Detection Module	22
	8.3 Eye Detection Module	22
	8.4 The Classification Module	24
	8.5 Predicting the image captured from the camera	25
9	**SAMPLE CODE**	27
10	**OUTPUT SCREENSHOT**	30
11	**RESULT AND ANALYSIS**	31
12	**FUTURE SCOPE AND APPLICATIONS**	33
13	**CONCLUSION**	35
	REFERENCES	36

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
5.1	Architecture of the System	12
6.1.1	Convolution Operation	16
6.1.2	Convolution with RGB images	17
6.1.3	Stride	17
6.1.1.2	Max Pooling	17
7.1	Systematic flow diagram	18
8.1	Haars Features	19
8.2	Extracting different features	20
10	Epoch Generated	21
11.1	Alert images	31
11.2	Drowsy faces	31
11.3	Alert images	31
11.4	Drowsy faces	31
11.5	Drowsy Output	32

LIST OF ABBREVIATIONS

ABBREVIATIONS

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
CV	Computer Vision
ANN	Artificial Neural Networks
CNN	Convolutional Neural Networks
SVM	Support Vector Machines
ReLU	Rectified Linear Unit
PPG	Photo Plethysmo Graphy
ROC	Receiver Operation Curve
EOG	Electrooculography
ECG	Electrocardiography
EEG	Electroencephalogram
EAR	Eye Aspect Ratio
HRV	Heart Rate Variability
FC	Fully Connected
ROI	Region Of Interest

Chapter 1

Introduction

Car accident is the major cause of death in which around 1.3 million people die every year. Majority of these accidents are caused because of distraction or the drowsiness of driver. The countless number of people drives for long distance every day and night on the highway. Drowsiness appears in situations of stress and fatigue in an unexpected and inopportune way, and it may be produced by sleep disorders, certain type of medications, and even, boredom situations, for example, driving for a long time. In this way, drowsiness produces danger situations and increases the probability that an accident occurs.
In this context, it is important to use new technologies to design and to build systems that will monitor drivers, and measure their level of attention throughout the whole driving process.

To prevent such accidents, our team has come up with a solution for this. In this system, a camera is used to record user's visual characteristics. We use face detection and CNN techniques and try to detect the drowsiness of driver, if he/she is drowsy then alarm will be generated. So that the driver will get cautious and take preventive measures. Driver drowsiness detection contributes to the decrease in the number of deaths occurring in traffic accident.

Traffic accidents due to human errors cause many deaths and injuries around the world. The major cause of these accidents is drowsiness of the driver due to sleeplessness or long driving hours. There is need for a system developed with the technologies that are available today which can overcome this situation. The aim of this system is to reduce the number of accidents by developing a model which can generate an alert if the driver is feeling drowsy so that the driver can become aware and take necessary actions.

Chapter 2

LITERATURE REVIEW

2.1. Drowsiness Detection System Utilizing Physiological Signals

Author: Trupti K. Dange, T. S. Yengatiwar.

Year of publication: 2013.

Keywords: EOG, ECG, EEG, HRV, SVM, driver drowsiness detection.

The Physiological parameters-based techniques detect drowsiness based on drivers' physical conditions such as heart rate, pulse rate, breathing rate, respiratory rate and body temperature, etc. These biological parameters are more reliable and accurate in drowsiness detection as they are concerned with what is happening with driver physically. Fatigue or drowsiness, change the physiological parameters such as a decrease in blood pressure, heart rate and body temperature, etc. Physiological parameters-based drowsiness detection systems detect these changes and alert the driver when he is in the state, near to sleep.

A list of physiological condition-based drowsiness detection system. These measures are invasive, so require electrodes to be directly placed on the driver's body.

1) EEG-BASED DRIVER FATIGUE DETECTION

The drivers' fatigue detection system using Electroencephalogram (EEG) signals is proposed to avoid the road accidents usually caused due to drivers' fatigue. The proposed method firstly finds the index related to different drowsiness levels. The system takes EEG signal as input which is calculated by a cheap single electrode neuro signal acquisition device. To evaluate the proposed method, data set for simulated car driver under the different levels of drowsiness is collected locally. And result shows that the proposed system can detect all subjects of tiredness.

2) WAVELET ANALYSIS OF HEART RATE VARIABILITY & SVM CLASSIFIER

Li and Chung [21] proposed the driver drowsiness detection that uses wavelet analysis of Heart Rate Variability (HRV) and Support Vector Machine (SVM) classifier. The basic purpose is to categorize the alert and drowsy drivers using the wavelet transform of HRV signals over short durations. The system firstly takes Photo Plethysmo Graphy (PPG) signal as input and divide it into 1-minute intervals and then verify two driving events using average percentage of eyelid closure over pupil over time (PERCLOS) measurement over the interval. Secondly, the system performs the feature extraction of HRV time series based on Fast Fourier Transform (FFT) and wavelet. A Receiver Operation Curve (ROC) and SVM classifier is used for feature extraction and

classification respectively. The analysis of ROC shows that the wavelet-based method gives improved results than the FFT-based method. Finally, the real time requirements for drowsiness detection, FFT and wavelet features are used to train the SVM classifier extracted from the HRV signals.

3) PULSE SENSOR METHOD

Mostly, previous studies focus on the physical conditions of drivers to detect drowsiness. That’s why Rahim detects the drowsy drivers using infrared heart-rate sensors or pulse sensors. The pulse sensor measures the heart pulse rate from drivers’ finger or hand. The sensor is attached with the finger or hand, detects the amount of blood flowing through the finger. Then amount of the blood’s oxygen is shown in the finger, which causes the infrared light to reflect off and to the transmitter. The sensor picks up the fluctuation of oxygen that are connected to the Arduino as microcontroller. Then, the heart pulse rate is visualizing by the software processing of HRV frequency domain. Experimental results show that LF/HF (Low to high frequency) ratio decreases as drivers go from the state of being awake to the drowsy and many road accidents can be avoided if an alert is sent on time.

4) WEARABLE DRIVER DROWSINESS DETECTION SYSTEM

Mobile based applications have been developed to detect the drowsiness of drivers. But mobile phones distract the drivers’ attention and may cause accident. To address the issue, Lenget proposed the wearable-type drowsiness detection system. The system uses self- designed wrist band consists of PPG signal and galvanic skin response sensor. The data collected from the sensors are delivered to the mobile device which acts as the main evaluating unit. The collected data are examined with the motion sensors that are built-in in the mobiles. Then five features are extracted from the data: heart rate, respiratory rate, stress level, pulse rate variability, and adjustment counter. The features are moreover used as the computation parameters to the SVM classifier to determine the drowsiness state. The experimental results show that the accuracy of the proposed system reaches up to
98.02 %. Mobile phone generates graphical and vibrational alarm to alert
the driver.

5) WIRELESS WEARABLES METHOD

To avoid the disastrous road accidents, Warwick proposed the idea for drowsiness detection system using wearable Bio sensor called Bio-harness. The system has two phases. In the first phase, the physiological data of driver is collected using bio-harness and then analyzes the data to find the key parameters like ECG, heart rate, posture and others related to the drowsiness. In the second phase, drowsiness detection algorithm will be designed and develop a mobile app to alert the drowsy drivers.

6) DRIVER FATIGUE DETECTION SYSTEM

Chellappa presents the Driver fatigue detection system. The basic of the system is to detect the drowsiness when the vehicle is in the motion. The system has three components: external hardware (sensors and camera), data processing module and alert

unit. Hardware unit communicates over the USB port with the rest of the system. Physiological and physical factors like pulse rate, yawning, closed eyes, blink duration and others are continuously monitored using somatic sensor. The processing module uses the combination of the factors to detect drowsiness. In the end, alert unit alerts the driver at multiple stages according to the severity of the symptoms.

7) HYBRID APPROACH UTILIZING PHYSIOLOGICAL FEATURES

To improve the performance of detection, Awais proposed the hybrid method which integrates the features of ECG and EEG. The method firstly extracts the time and frequency domain features like time domain statistical descriptors, complexity measures and power spectral measures from EEG. Then using ECG, features like heart rate, HRV, low frequency, high frequency and LF/HF ratio. After that, subjective sleepiness is measured to study its relationship with drowsiness. To select only statistically significant features, t-tests is used that can differentiate between the drowsy and alert. The features extracted from ECG and EEG are integrated to study the improvements in the performance using SVM. The other main contribution is to study the channel reduction and its impact on the performance of detection. The method measures the differences between the drowsy and alert state from physiological data collected from the

driving simulated-based study. Monotonous driving environment is used to induce the drowsiness in the participants. The proposed method demonstrated that combining ECG and EEG improves the performance of system in differentiating the drowsy and alert states, instead of using them alone. The analysis of channel reduction confirms that the accuracy level reaches to 80% by just combining the

ECG and EEG. The performance of the system indicates that the proposed system is feasible for practical drowsiness detection system.

2.2. Drowsiness Detection with OpenCV (Using Eye Aspect Ratio)

Author: Adrian Rosebrock.

Year of publication: 2017.

Keywords: EAR, SVM, eye blink detection.

A real-time algorithm to detect eye blinks in a video sequence from a standard camera is proposed. Recent landmark detectors, trained on in-the wild datasets exhibit excellent robustness against a head orientation with respect to a camera, varying illumination and facial expressions. We show that the landmarks are detected precisely enough to reliably estimate the level of the eye opening. The proposed algorithm therefore estimates the landmark positions, extracts a single scalar quantity – eye aspect ratio (EAR) – characterizing the eye opening in each frame.

Many methods have been proposed to automatically detect eye blinks in a video sequence. Several methods are based on motion estimation in the eye region. Typically, the face and eyes are detected by a Viola-Jones type detector. Next, motion in the eye area is estimated from optical flow, by sparse tracking, or by frame-to-frame intensity differencing and adaptive thresholding. Finally, a decision is made whether the eyes are or are not covered by eyelids.

Nowadays, robust real-time facial landmark detectors that capture most of the characteristic points on a human face image, including eye corners and eyelids, are available, Most of the state-of-the-art landmark detectors formulate a regression problem, where a mapping from an image into landmark positions or into other landmark parametrization is learned. These modern landmark detectors are trained on "in-the-wild datasets" and they are thus robust to varying illumination, various facial expressions, and moderate non-frontal head rotations

Proposed method

The eye blink is a fast closing and reopening of a human eye. Each individual has a little bit different pattern of blinks. The pattern differs in the speed of closing and opening, a degree of squeezing the eye and in a blink duration. The eye blink lasts approximately 100-400 ms. We propose to exploit state-of-the-art facial landmark detectors to localize the eyes and eyelid contours. From the landmarks detected in the image, we derive the eye aspect ratio (EAR) that is used as an estimate of the eye opening state. Since the per frame EAR may not necessarily recognize the eye blinks correctly, a classifier that takes a larger temporal window of a frame into account is trained. The EAR is mostly constant when an eye is open and is getting close to zero while closing an eye.

A real-time eye blink detection algorithm was presented. We quantitatively demonstrated that regression-based facial landmark detectors are precise enough to reliably estimate a level of eye openness. While they are robust to low image quality (low image resolution in a large extent) and in-the-wild phenomena as non-frontality, bad illumination, facial expressions, etc.
The proposed SVM method that uses a temporal window of the eye aspect ratio (EAR), outperforms the EAR thresholding.

$$\text{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

2.3. Real Time Driver Fatigue Detection Based on SVM Algorithm

Authors: Burcu Kir Savas, Yasar Becerkli.

Year of publication: 2018.

Keywords: fatigue detection , SVM , driving safety , video processing

PROPOSED SYSTEM

In this study, SVM based driver fatigue prediction system is proposed to increase driver safety. The proposed system has five stages: PERCLOS, count of yawn, internal zone of the mouth opening, count of eye blinking and head detection to extract attributes from video recordings. The classification stage is done with Support Vector Machine (SVM). While the YawDD dataset is used during the training phase of the classification, real-time video recordings are used during the test phase.

SVM is a classification algorithm separating data items. This algorithm proposed by Vladimir N. Vapnik based on statistical learning theory. SVM, one of the machine learning methods, is widely used in the field of pattern recognition. The main purpose of the SVM is to find the best hyperplane to distinguish the data given as two-class or multiclass. The study was carried out in two classes. Whereas label 0 means that the driver is tired, label 1 means the driver is non-tired. Thus, it is aimed to distinguish tired drivers (driver fatigue) from non-tired drivers.

In training driving simulation experiments, we had 10 volunteers (5 men and 5 women) whose ages were between 18-30. All the volunteers drove training simulation during the experiments. Attributions obtained in real time during the driving when fatigue detection system was working are indicated. The results of driver fatigue detection were classified using SVM classification algorithm. The total 713 datas line each of which has five stage features were sampled from 10 volunteers. All the data in the study are divided into two groups: 80% training data set (568 data set) and 20% test data set (145 data set). In the study, cross validation was selected as 10.

Measurements are as follows:

- TP means that a real fatigue is detected;
- TN means that a non-fatigue situation is correctly detected as non-fatigue by the system;
- FP means that a non-fatigue situation is incorrectly detected as real fatigue;
- FN means that a real fatigue situation is incorrectly detected as non-fatigue.

The accuracy of driver fatigue calculated as Accuracy= (TP+TN)/(TP+TN+FP+FN)

As a result of the study, a system was designed to investigate the effects of fatigue and

insomnia on drivers physical transient behaviors, and to simulate drowsy drivers as a result of research. In this system, behavioral detection model is used for detecting fatigue drivers. The proposed system has five stages: PERCLOS, count of yawn, internal zone of the mouth opening, count of eye blinking and head detection to extract attributes from video recordings. As a result, the system is classified as SVM in two classes (not fatigue / fatigue). In this study a Real Time Driver Fatigue Detection SVM Based on SVM Algorithm is presented whose test] results showed that the accuracy rate of fatigue detection is up to 97.93%

2.4. Deep CNN: A Machine Learning Approach for Driver Drowsiness Detection Based on Eye State

Authors: Venkata Rami Reddy Chirra, Srinivasulu Reddy Uyyala and Venkata

Krishna Kishore Kolli.

Year of publication: 2019.

Keywords: Deep CNN, Feature extraction

PROPOSED DEEP CNN BASED DROWSINESS DETECTION SYSTEM

Proposed system algorithm

(1) Viola-jones face detection algorithm is used to detect the face the images and given as input to Viola-jones eye detection algorithm
(2) Once the face is detected, Viola-jones eye detection algorithm is used to extract the eye region from the facial images and given as input to CNN.
(3) CNN with four convolutional layers is used to extract the deep features and those features are passed to fully connected layer.
(4) Soft max layer in CNN classify the images in to sleepy or non-sleepy images.

Face detection and eye region extraction

Whole face region may not be required to detect the drowsiness but only eyes region is enough for detecting drowsiness. At first step by using the Viola-jones face detection algorithm face is detected from the images. Once the face is detected, Viola-jones eye detection algorithm is used to extract the eye region from the facial images. it is the first algorithm used for face
detection. For the face detection the Viola-Jones algorithm having three techniques those are Haar-like features, Ada boost and Cascade classifier. In this work, Viola-Jones object detection algorithm with Haar cascade classifier was used and implemented using Open CV with python. Haar cascade classifier uses Haar features for detecting the face from images.

Feature extraction and classification

Feature extraction is one type of dimensionality reduction where useful parts of an image represented as a feature vector.

Convolutional neural network

Convolutional neural network (CNN) is used in the proposed system for detection of driver drowsiness. Since a feature vector is needed for each drowsy image to compare with existing features in a database to detect either drowsy or not. Usually CNNs requires fixed size images as input so preprocessing is required. The preprocessing includes extracting the key frames from video based on temporal changes and store in database. From these stored images, feature vectors are generated in convolution layers of CNN. These feature vectors are then used for the detecting the driver drowsiness. CNN have layers like convolutional layers, pooling (max, min and average) layers, ReLU layer and fully connected layer. Convolution layer is having kernels (filters) and each kernel having width, depth and height. This layer produces the feature maps as a result of calculating the scalar product between the kernels and local regions of image. CNN uses pooling layers

(Max or Average) to minimize the size of the feature maps to speed up calculations. In this layer, input image is divided into different regions then operations are performed on each region. In Max Pooling, a maximum value is selected for each region and places it in the corresponding place in the output. ReLU (Rectified Linear Units) is a nonlinear layer. The ReLU layer applies the max function on all the values in the input data and changes all the negative values to zero. The following equation shows the ReLU activation function.

Chapter 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The existing system used Support Vector Machine (SVM) for classifying the face (if the eyes ae closed or not) as drowsy or not drowsy. To separate the two classes of data points, there are many possible hyperplanes that could be chosen. The objective of SVM is to find a plane that has the maximum margin, i.e., the maximum distance between data points of both classes.

3.1.1 Disadvantages

- SVM does not perform very well when the data set has more noise i. e target classes are overlapping.
- It requires more time to process

3.2 PROPOSED SYSTEM

In this system, instead of Support Vector Machine (SVM) we use a Classification Model based on Convolutional Neural Networks (CNN). Deep Learning is concerned with algorithms inspired by the structure and function of the brain called artificial neural networks. Convolutional Neural Networks are a type of Artificial Neural Networks which are widely used for image classification and even Multi-Class classification of images.
Convolution layers in a CNN consist of a set of learnable filters. During forward propagation, we slide each filter across the whole input step by step where each step is called stride.
We propose CNN since the accuracy of the system is improved by using a CNN. The driver’s face is continuously captured using a camera. The image frames are extracted and by face detection, the face of the driver is detected.
A classification model is built based on CNN to classify the face as drowsy / not drowsy.

3.2.1 Advantages

- This CNN base system provide High Accuracy.
- Easy of use.
- Work well in low lighting condition

Chapter 4

SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENTS

- 1GB Hard Disk Space
- 4GB RAM Memory
- Web Camera
- Speaker

4.2 SOFTWARE REQUIREMENTS

- Windows OS
- Python 3.0 or Greater
- IDE : VS Code
- Open CV
- Keras
- Tensorflow
- Pygame

Chapter 5

DESIGN OF THE SYSTEM

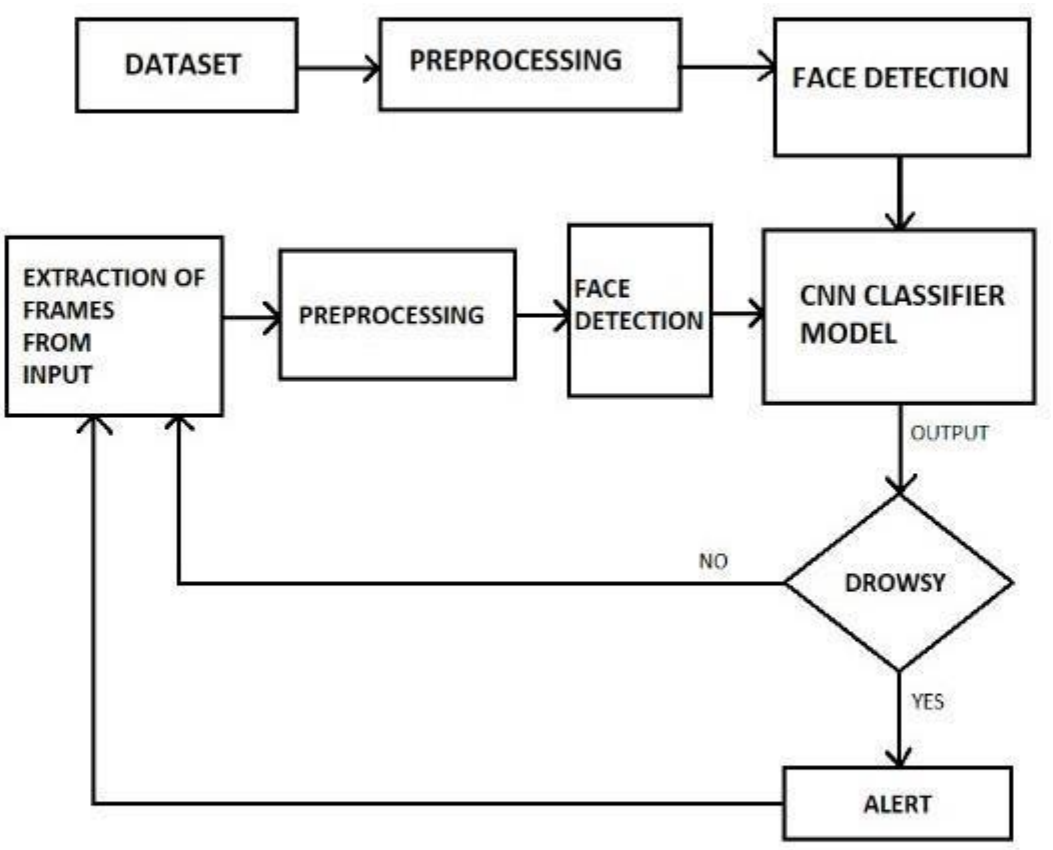


Fig 5.1. Architecture of the System

The model we used is built with Keras using **Convolutional Neural Networks (CNN).** A convolutional neural network is a special type of deep neural network which performs extremely well for image classification purposes. A CNN basically consists of an input layer, an output layer and a hidden layer which can have multiple numbers of layers. A convolution operation is performed on these layers using a filter that performs 2D matrix multiplication on the layer and filter.

The system uses a web camera that points directly towards the driver's face and monitors the driver's head movements in order to detect fatigue. In such a case when fatigue is detected, a warning signal is issued to alert the driver.

Chapter 6

PROPOSED METHODOLOGY

Driver exhaustion and drowsiness are significant contributors to various automobile accidents. In the field of accident prevention systems, designing and maintaining technology that can effectively identify or avoid drowsiness at the wheel and warn the driver before a collision is a major challenge. We use OpenCV to take images from a webcam and these images given to a deep learning algorithm that can tell whether someone's eyes are closed or opened. In this case, we are looking for the persons face and eyes.

STEP1: IMAGE IS TAKEN AS INPUT FROM CAMERA.

We'll use a camera to capture photographs as input. But, in order to gain access to the webcam, we created an endless loop that captures each frame. We employ the cv2 method given by OpenCV. VideoCapture(0) (cap) is used to access the camera and
capture the object. With cap.read(), each frame is read, and then image is saved in a variable.

STEP 2:CREATE A ROI BY DETECTING A FACE IN THE PICTURE.

To segment the face in the captured image, we first converted it to gray scale because, the OpenCV object detection algorithm only accepts grayscale images as input. To detect the objects, we don't need colour detail. We use the Haar cascade classifier to detect the face. The classifier face= cv2.Cas is set with this section. For (x,y,w,h)infaces,we use cv2.rectangle(frame, (x,y), (x+w, y+h), (100,100,100), 1

STEP 3: USE THE ROI TO FIND THE EYES AND FEED THEM TO THE CLASSIFIER.

The technique for detecting eyes is the same as for detecting ears. Cascade classifier is used in left and right eyes.Then, use left_eye=leye.detectMultiScale(gray) to detect the eyes. We extracted
only the details of eyes from the captured image. This can be done by first removingthe eye's boundary box and then using this code to remove the eye image from the picture.
l_eye = frame[y : y+h, x : x+w]
This information is given to CNN, which decides whether the eyes are closed or not. The right eye also detected in the above manner.

STEP 4 – THE CLASSIFIER WILL DETERMINE WHETHER OR NOT THE EYES ARE OPEN.

The eye status is predicted using a CNN classifier to feed the image into the model, since the model requires the proper measurements to begin with. We begin by converting the colour picture to grayscale.
r_eye=cv2.cvtColor(r_eye,cv2.COLOR_BGR2GRAY). Then, since the model is trained on images with a resolution of 24*24 pixels, We resize the image to 24*24pixels.
cv2.resize (r_eye, (24,24)).
For better convergence, the date is normalized.
r_eye = r_eye/255
The model is loaded using model=load_model(„models/cnnCat2.h5‟)
Now, each eye is predicted with the proposed model.
lpred=model.predict_classes(l_eye)
If lpred[0] = 1, it means that eyes are open, if lpred[0] = 0 then, it means that eyes are closed.

STEP 5: SCORE CALCULATION.

The score is essentially a number that we'll use to figure out how long the individual has been closed-eyed. As a consequence, if both eyes are closed, we will begin to raise the score, but if both eyes are open, we will decrease the score. We're using the cv2.putText() function to draw the result on the screen, which displays the status of the
driver or a person.
cv2.putTxt(frame,"Open",(10,height20), font,1,(255,255,255),1,cv2.LINE_AA)
A criterion is established, for example, if the score exceeds 15, it indicates that the person's eyes have been closed for an extended amount of time. Then the alarm turned on

6.1.CONVOLUTIONAL NEURAL NETWORK

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

The term “Convolution” in CNN denotes the mathematical function of convolution which is a special kind of linear operation wherein two functions are multiplied to produce a third function which expresses how the shape of one function is modified by the other. In simple terms, two images which can be represented as matrices are multiplied to give an output that is used to extract features from the image.

There are two main parts to CNN architecture:

- A convolution tool that separates and identifies the various features of the image for analysis in a process called as Feature Extraction
- A fully connected layer that utilizes the output from the convolution process and predicts the class of the image based on the features extracted in previous stages.

There are three types of layers that make up the CNN which are the convolutional layers, pooling layers, and fully-connected (FC) layers. When these layers are stacked, CNN architecture will be formed. In addition to these three layers, there are two more important parameters which are the dropout layer and the activation function which are defined below.

6.1.1. CONVOLUTIONAL LAYER

Convolutional layers are the major building blocks used in convolutional neural networks. This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size MxM. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter (MxM). If the filter is designed to detect a specific type of feature in the input, then the application of that filter systematically across the entire input image allows the filter an opportunity to discover that feature anywhere in the image.

The output is termed as the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image.

THE CONVOLUTION OPERATION

We analyze the influence of nearby pixels by using something called a filter. We move this across the image from top left to bottom right. When building the network, we randomly specify values for the filters, which then continuously update themselves as the network is trained. After the filters have passed over the image, a feature map is generated for each filter. These are then taken through an activation function, which decides whether a certain feature is present at a given location in the image.

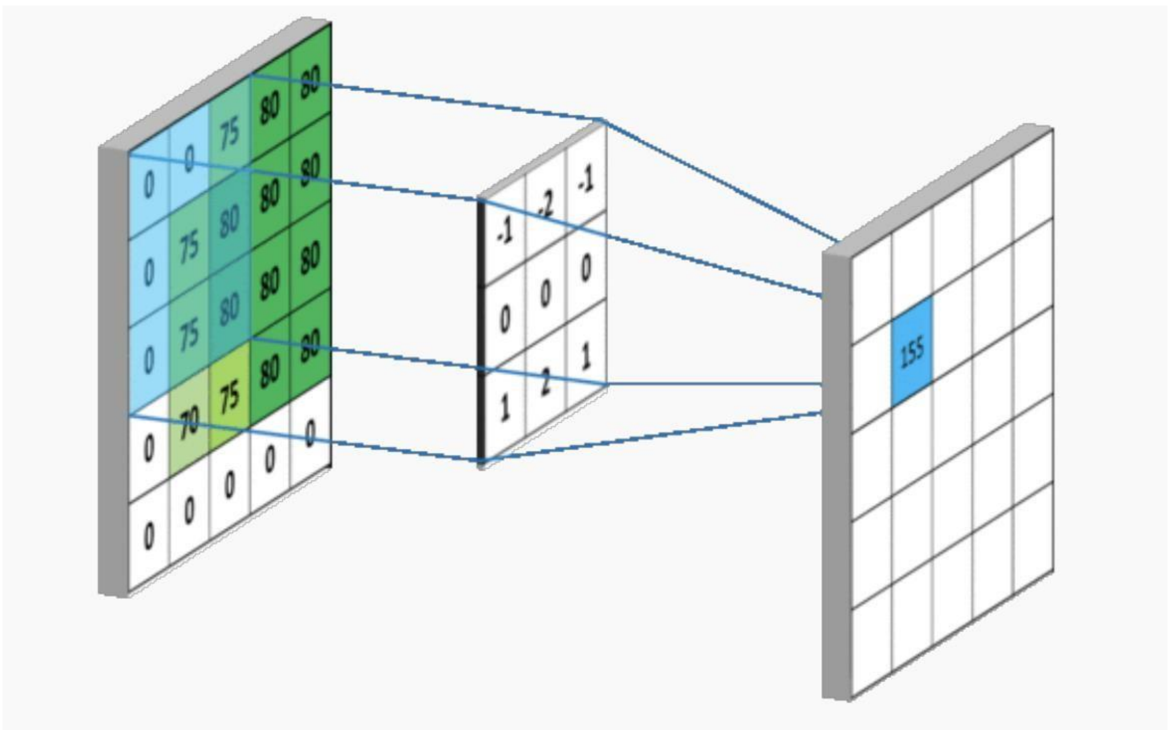


Fig. 6.1.1. Convolution Operation

We have colour images which have 3 channels i.e., Red, Green and Blue. Each filter is convolved with the entirety of the 3D input cube but generates a 2D feature map. Because we have multiple filters, we end up with a 3D output: one 2D feature map per filter. Convolving the image with a filter produces a feature map that highlights the presence of a given feature in the image.

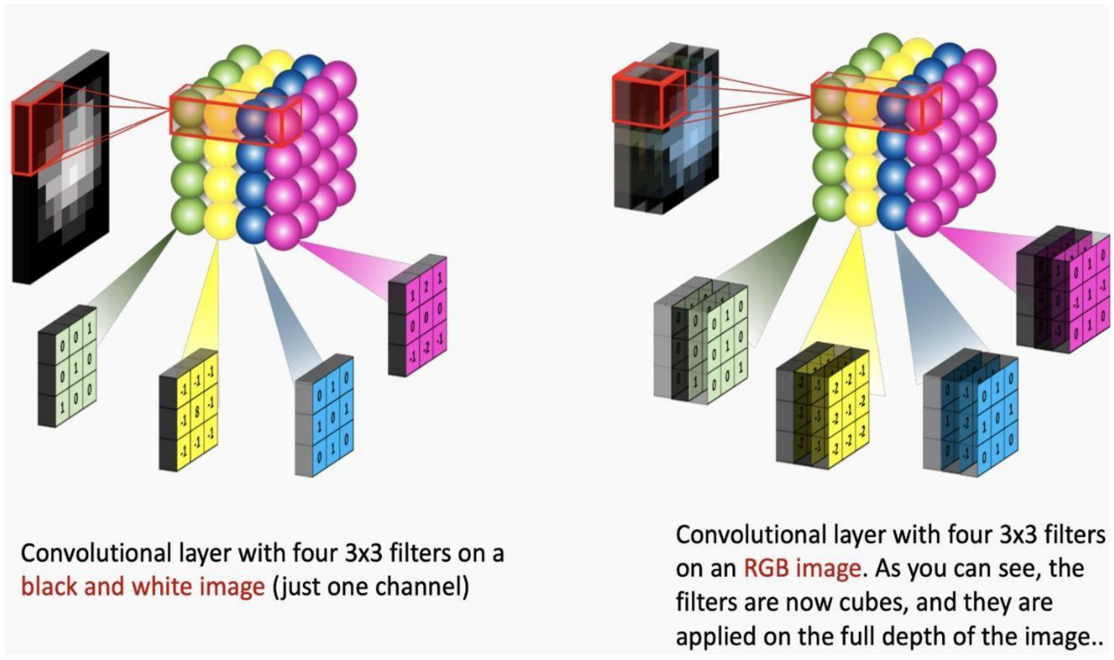

Convolutional layer with four 3x3 filters on a black and white image (just one channel)
Convolutional layer with four 3x3 filters on an RGB image. As you can see, the filters are now cubes, and they are applied on the full depth of the image..

Fig. 6.1.2. Convolution with RGB images

STRIDE

Stride is the number of pixels shifts over the input matrix. The filter moves to the right with a certain Stride Value till it parses the complete width. Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed. When the stride is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and so on. The below figure shows convolution would work with a stride of 2.

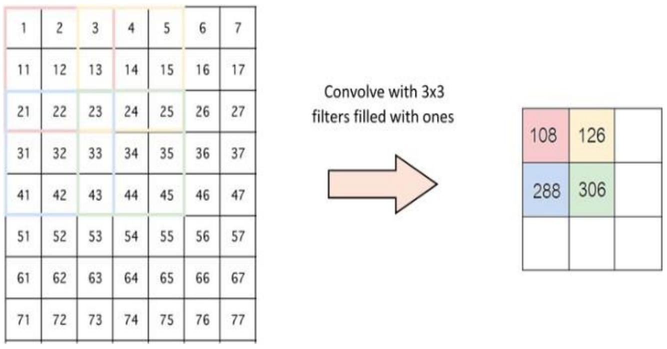


Fig. 6.1.3. Stride

6.1.1.1.POOLING LAYER

In most cases, a Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature map. Depending upon method used, there are several types of Pooling operations.

In Max Pooling, the largest element is taken from feature map. Average Pooling calculates the average of the elements in a predefined sized Image section. The total sum of the elements in the predefined section is computed in Sum Pooling. The Pooling Layer usually serves as a bridge between the Convolutional Layer and the FC Layer.

6.1.1.2.MAX POOLING

Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

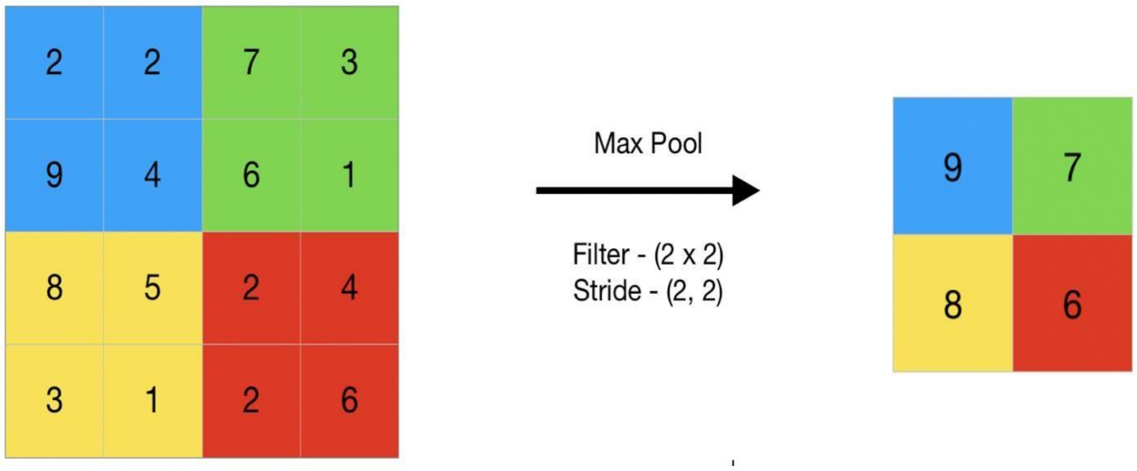


Fig. 6.1.1.2. Max Pooling

6.1.1.3.FULLY CONNECTED LAYER

The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture.

In this, the input image from the previous layers are flattened and fed to the FC layer. The flattened vector then undergoes few more FC layers where the mathematical functions operations usually take place. In this stage, the classification process begins to take place.

Chapter 7

SYSTEM DESIGN

7.1. SYSTEMATIC FLOW DIAGRAM

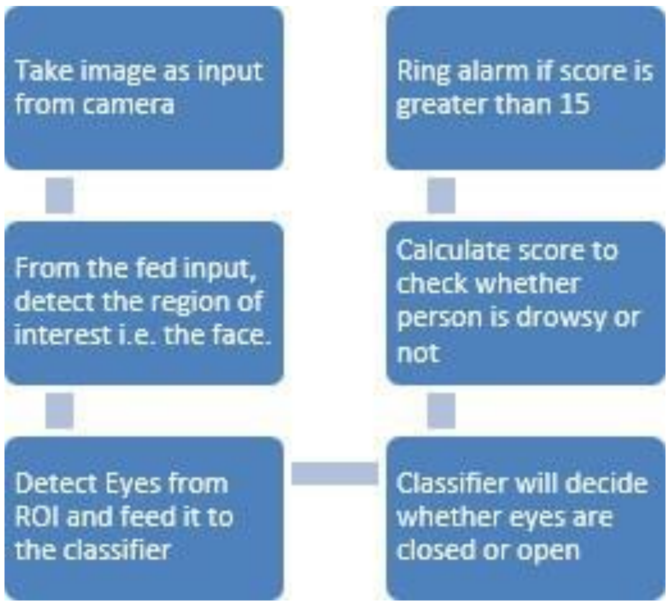

Take image as input from camera
From the fed input, detect the region of interest i.e. the face.
Detect Eyes from ROI and feed it to the classifier
Classifier will decide whether eyes are closed or open
Calculate score to check whether person is drowsy or not
Ring alarm if score is greater than 15

Fig. 7.1. Systematic flow diagram

7.1.1. With the help of a camera, each frame is captured continuously, for this, it is ideal to create and infinite loop by the method provided by OpenCV i.e., cv2.VideoCapture(0), each frame is captured and stored in a variable named frame.

7.1.2. Next, the image is converted into greyscale, as the OpenCV algorithm takes only grey images for processing. The colour information is irrelevant anyway. Haar Cascade classifier is user to detect faces in the images captured. The classifier returns an array of detections with x, y coordinates height and width of the boundary of the box of the object. Each frame is iterated and boundary box for each face is detected.

7.1.3. Now, the same procedure as above is applied to extract eyes from the face. Cascade Classifier is used to first detect left eye and then the right eye. The data of left eye is fed to l_eye variable and the data of right eye is fed to r_eye variable. The data of l_eye is fed to CNN classifier which will protect whether the eye is closed or not, similar process is applied to r_eye.

7.1.4. Before feeding out data to CNN classifier, certain changes to be made due to its need of accurate measurements in the beginning. First, image is converted to greyscale and then resized to 24*24 because this is the size that the model is trained to handle. Then the data is normalized for better convergence. Now, the image is ready to be fed to the CNN classifier, after feeding the image to the classifier, predictions are made for both left and right eye.

7.1.5. A score criterion is set to determine the threshold for detection. In this proposed model, the score is set to 15. Once the eyes are closed, the score will start increasing, after it attains the threshold value, the alarm goes off. The python library pyplay is used along with the function sound.play() for the alarm to ring.

Chapter 8

SYSTEM MODULES

8.1.COLLECTING THE DATASET

The first step is collecting the dataset. To work with machine learning projects, we need a huge amount of data, because, without the data, one cannot train ML/AI models. Collecting and preparing the dataset is one of the most crucial parts while creating an ML/AI project.
A dataset called Drowsiness_dataset from Kaggle which contains images with open eyes and close eyes. This dataset also includes images with the people wearing spectacles.
Another dataset from Github is considered in which the images of 3 people are recorded while driving. This dataset contains 3 sets of images which are alert images, images with closed eyes and. The images of closed eyes are combined as drowsy images.

8.2.FACE DETECTION MODULE

This module takes input from the camera and tries to detect a face in the video input. The detection of the face is achieved through the Haar classifiers mainly, the Frontal face cascade classifier. The face is detected in a rectangle format and converted to grayscale image and stored in the memory which can be used for training the model.

Instead of giving the entire image to classification model, only the face region is extracted and given to the model since the background and other portion of the image is unnecessary. For this, we use Face detection which is a computer vision technology that locates human faces in digital images. Haar Cascades Algorithm, also known Viola-Jones Algorithm is used for Face detection which was proposed by Paul Viola and Michael Jones in their 2001 paper.

8.3.EYE DETECTION MODULE

Since the model works on building a detection system for drowsiness we need to focus on the eyes to detect drowsiness. The eyes are detected through the video input by implementing a haar classifier namely **Haar Cascade Eye Classifier**. The eyes are detected in rectangular formats.

Working of the algorithm:

This algorithm uses Haar features to extract objects.

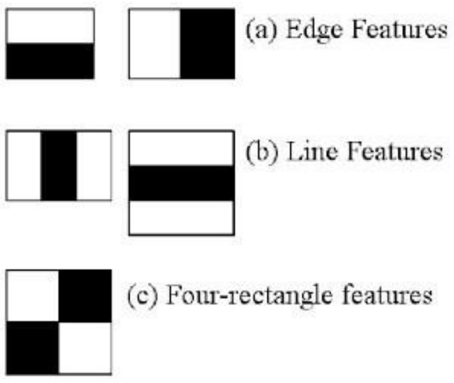


Fig. 8.1. Haar Features

The algorithm applies the features on windows of the image, gradually changing the window size after each turn. If a particular window is not classified as a face, that window is not processed in further steps.
Instead of applying all the features, it makes use of cascade of classifiers concept. The features are grouped into different stages of classifiers and applied one-by-one. Only if a window passes a stage, the next stage is applied on that or else the window will be discarded and no longer considered as face region. A face region is the one which passes all the stages.

The openCV detectmultiscale() returns the starting coordinates, width and height of the detected face, using which the face region is extracted from the corresponding image and resized to a fixed size 100x100.
We append all the resized images to a list and the corresponding image labels to another list. We convert both the lists to numpy arrays and we give these arrays to the Classification model.

The parameters used in detectmultiscale() method are image, scaleFactor and minNeighbors. The scaleFactor parameter specifies how much the image size is reduced at each image scale. The minNeighbor parameter specifies how many neighbors each candidate rectangle should have to retain in. This parameter affects the quality of the detected faces.

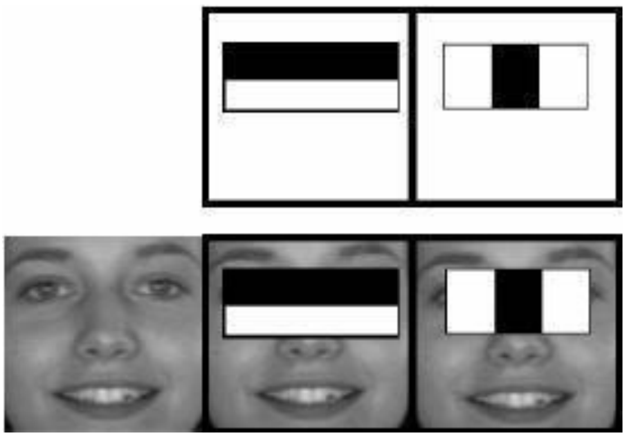


Fig 8.2. Extracting different features

8.4.THE CLASSIFICATION MODULE

In this module, we build a Deep Learning Binary Classification model to classify the images as alert / drowsy. We used sequential model.
In machine learning, classification refers to a predictive problem where a class label is predicted for a given example of input data. A model will use the training dataset and will calculate how to best map examples of input data to specific class labels. Binary classification refers to those classification tasks that have two class labels.

Convolutional Neural Networks, or CNNs, were designed to map image data to an output variable. The benefit of using CNNs is their ability to develop an internal representation of a two-dimensional image.

A convolution is essentially sliding a filter over the input. Each convolutional layer contains a series of filters known as convolutional kernels. The filter is a matrix of integers that are used on a subset of the input pixel values. Each pixel is multiplied by the corresponding value in the kernel, then the result is summed up for a single value for simplicity representing a grid cell, like a pixel, in the output channel/feature map.

The CNN model architecture consists of the following layer
- Convolutional layer; 32 nodes, kernel size 3
- Convolutional layer; 32 nodes, kernel size 3
- Convolutional layer; 64 nodes, kernel size 3
- Fully connected layer; 128 nodes

The final layer is also a fully connected layer with 2 nodes. In all the layers, a Relu activation function is used except the output layer in which we used Softmax.

Training:

We shuffle the data and split all the available images into train-data(80%) and test- data(20%). We run 20 epochs during training. While training the model, some part of the training data is used as validation data. We use 10% of the training data for validation. The validation data is used to evaluate the performance of the model at the end of each epoch but not used to train the model.

We use checkpoints in our model to save the state of the model each time an improvement is observed during training. The checkpoint stores the weights of the model. The weights will be updated after each improvement and the best model will be saved when the training is completed.

Evaluation:

After training the model, it's performance should be measured. There are different metrics to evaluate a Classification model like Loss, Accuracy, Precision and Recall.

We use Matplotlib to plot the graphs of different metrics of the model.

Testing with test data:

Testing is used to evaluate the generalizing ability of the model by giving unseen data to it. The data is previously split into training data and test data. The 20% of the data which is not known to the model is given to the model and the model will predict the classes for the test data. We evaluate the performance of the model by finding loss and accuracy with the test data. We also find the Precision and Recall values for the model.

8.5.PREDICTING THE IMAGES CAPTURED FROM THE CAMERA

After the model is trained with the given dataset, we can use the model to predict the class of the images which are captured from the camera.

We use OpenCV to capture the images from the camera. We continuously capture image frames from the camera. The same preprocessing steps which are applied on the dataset are applied on each frame captured i.e., detecting the face from the image frame, extract the Region of Interest and then resizing the Region of Interest to a fixed size (100x100). Then we convert the images into array format to give as input to the model.

Then, we can give a set of images to the trained CNN Classification model to predict the labels for the images.

If the driver is feeling drowsy, a voice alert is generated. An audio file is played using the playsound module in python.

Chapter 9

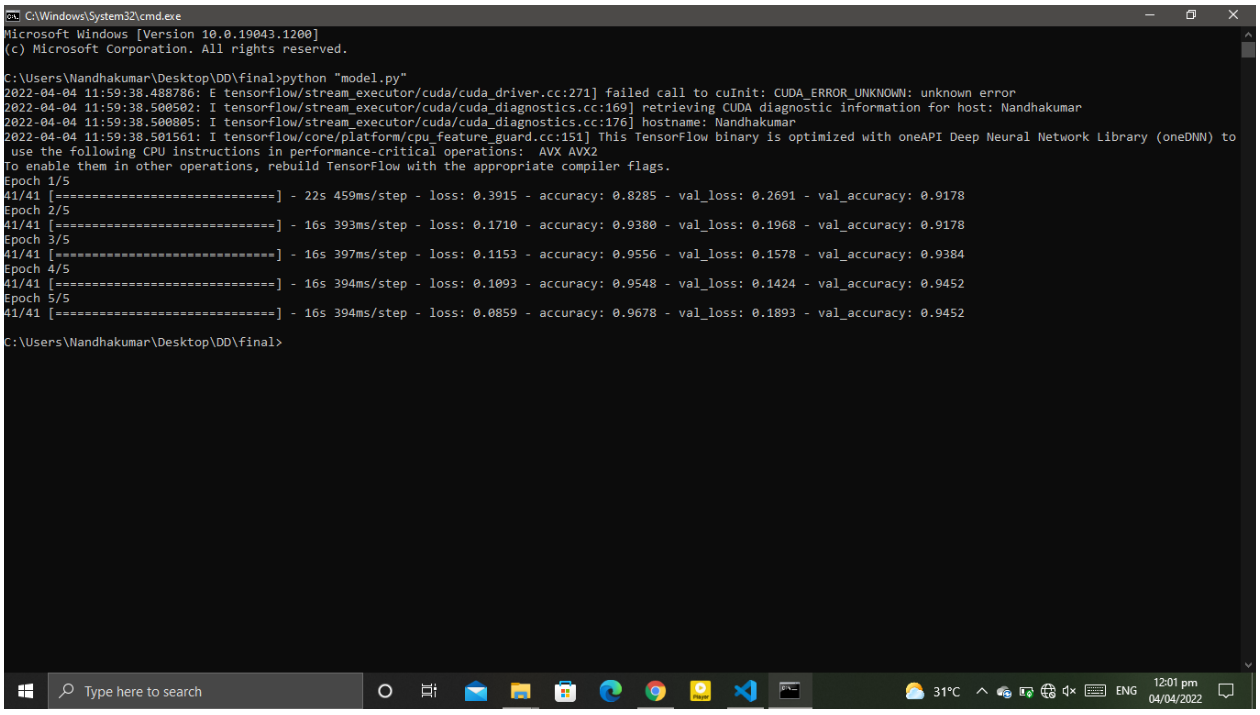
SAMPLE CODE

Drowsiness Detecion :

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


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

face = cv2.CascadeClassifier('haar cascade files\haarcascade_frontalface_alt.xml')
leye = cv2.CascadeClassifier('haar cascade files\haarcascade_lefteye_2splits.xml')
reye = cv2.CascadeClassifier('haar cascade files\haarcascade_righteye_2splits.xml')



lbl=['Close','Open']

model = load_model('models/cnncat2.h5')
path = os.getcwd()
cap = cv2.VideoCapture(0)
font = cv2.FONT_HERSHEY_COMPLEX_SMALL
count=0
score=0
thicc=2
rpred=[99]
lpred=[99]

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

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = 
face.detectMultiScale(gray,minNeighbors=5,scaleFactor=1.1,minSize=(25,25))
    left_eye = leye.detectMultiScale(gray)
    right_eye =  reye.detectMultiScale(gray)

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

    for (x,y,w,h) in faces:
        cv2.rectangle(frame, (x,y) , (x+w,y+h) , (100,100,100) , 1 )

    for (x,y,w,h) in right_eye:
        r_eye=frame[y:y+h,x:x+w]
        count=count+1
        r_eye = cv2.cvtColor(r_eye,cv2.COLOR_BGR2GRAY)
        r_eye = cv2.resize(r_eye,(24,24))
        r_eye= r_eye/255
        r_eye=  r_eye.reshape(24,24,-1)
        r_eye = np.expand_dims(r_eye,axis=0)
        rpred = np.argmax(model.predict(r_eye), axis=-1)
        if(rpred[0]==1):
            lbl='Open'
        if(rpred[0]==0):
            lbl='Closed'
        break

    for (x,y,w,h) in left_eye:
        l_eye=frame[y:y+h,x:x+w]
        count=count+1
        l_eye = cv2.cvtColor(l_eye,cv2.COLOR_BGR2GRAY)
        l_eye = cv2.resize(l_eye,(24,24))
        l_eye= l_eye/255
        l_eye=l_eye.reshape(24,24,-1)
        l_eye = np.expand_dims(l_eye,axis=0)
        lpred = np.argmax(model.predict(l_eye), axis=-1)
        if(lpred[0]==1):
            lbl='Open'
        if(lpred[0]==0):
            lbl='Closed'
        break
    if(rpred[0]==0 and lpred[0]==0):
        score=score+1
        cv2.putText(frame,"Closed",(10,height-20), font,
1,(255,255,255),1,cv2.LINE_AA)
    # if(rpred[0]==1 or lpred[0]==1):
    else:
        score=score-1
        cv2.putText(frame,"Open",(10,height-20), font,
1,(255,255,255),1,cv2.LINE_AA)


    if(score<0):
        score=0
    cv2.putText(frame,'Score:'+str(score),(100,height-20), font,
1,(255,255,255),1,cv2.LINE_AA)
    if(score>15):
        #person is feeling sleepy so we beep the alarm
        cv2.imwrite(os.path.join(path,'image.jpg'),frame)
        try:
            sound.play()

        except:  # isplaying = False
            pass
        if(thicc<16):
            thicc= thicc+2
        else:
            thicc=thicc-2
            if(thicc<2):
                thicc=2
        cv2.rectangle(frame,(0,0),(width,height),(0,0,255),thicc)
    cv2.imshow('frame',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

Chapter 10

OUTPUT SCREENSHOTS



```
Microsoft Windows [Version 10.0.19043.1200]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Nandhakumar\Desktop\DD\final>python "model.py"
2022-04-04 11:59:38.488786: E tensorflow/stream_executor/cuda/cuda_driver.cc:271] failed call to cuInit: CUDA_ERROR_UNKNOWN: unknown error
2022-04-04 11:59:38.500502: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic information for host: Nandhakumar
2022-04-04 11:59:38.500805: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: Nandhakumar
2022-04-04 11:59:38.501561: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to
 use the following CPU instructions in performance-critical operations:  AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Epoch 1/5
41/41 [==============================] - 22s 459ms/step - loss: 0.3915 - accuracy: 0.8285 - val_loss: 0.2691 - val_accuracy: 0.9178
Epoch 2/5
41/41 [==============================] - 16s 393ms/step - loss: 0.1710 - accuracy: 0.9380 - val_loss: 0.1968 - val_accuracy: 0.9178
Epoch 3/5
41/41 [==============================] - 16s 397ms/step - loss: 0.1153 - accuracy: 0.9556 - val_loss: 0.1578 - val_accuracy: 0.9384
Epoch 4/5
41/41 [==============================] - 16s 394ms/step - loss: 0.1093 - accuracy: 0.9548 - val_loss: 0.1424 - val_accuracy: 0.9452
Epoch 5/5
41/41 [==============================] - 16s 394ms/step - loss: 0.0859 - accuracy: 0.9678 - val_loss: 0.1893 - val_accuracy: 0.9452

C:\Users\Nandhakumar\Desktop\DD\final>
```

Fig 10. Epoch Generated

Chapter 11

RESULTS & ANALYSIS

The input is an image which contains a human face. The following are some of the images from our dataset:

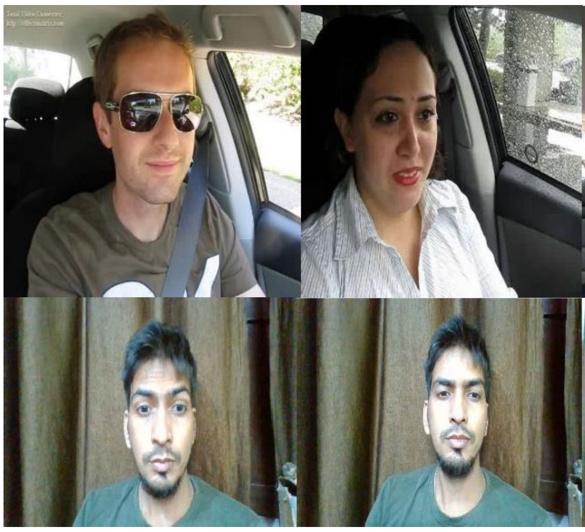
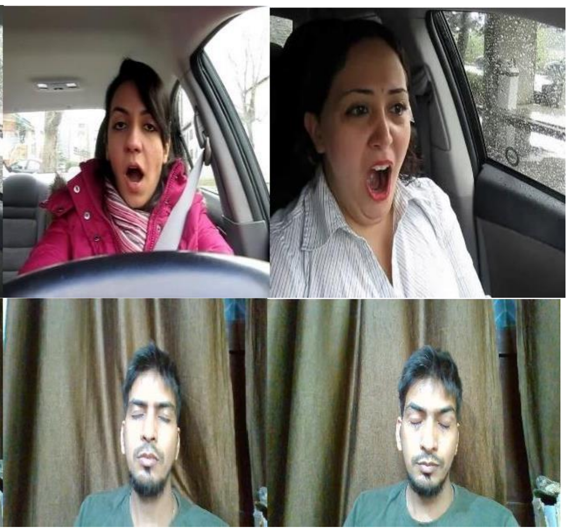



Fig. 11.1. Alert images **Fig. 11.2. Drowsy images**

Detected Faces:

The following are the images after performing face detection and resizing of the images:

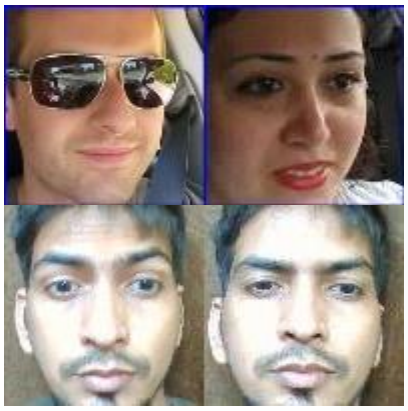
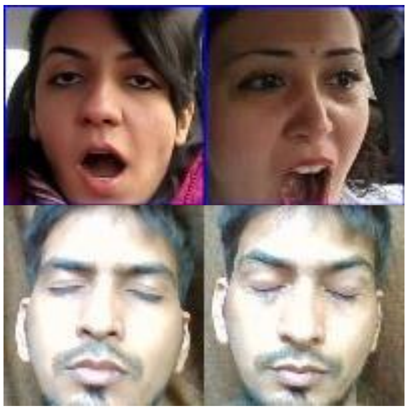



Fig. 11.3. Alert faces **Fig. 11.4. Drowsy faces**

Results of frames captured from camera:

Frames classified as drowsy:

Here the input will be the continuous stream of video, if the driver seems to be detected as drowsy then the detected face will be highlighted using a red colored rectangle and system gives an alert. The alert will be in the form of a sound (beep sound). The aim is to make the driver aware that he/she is drowsy with that sound. The drowsiness is detected by using the well trained CNN model.

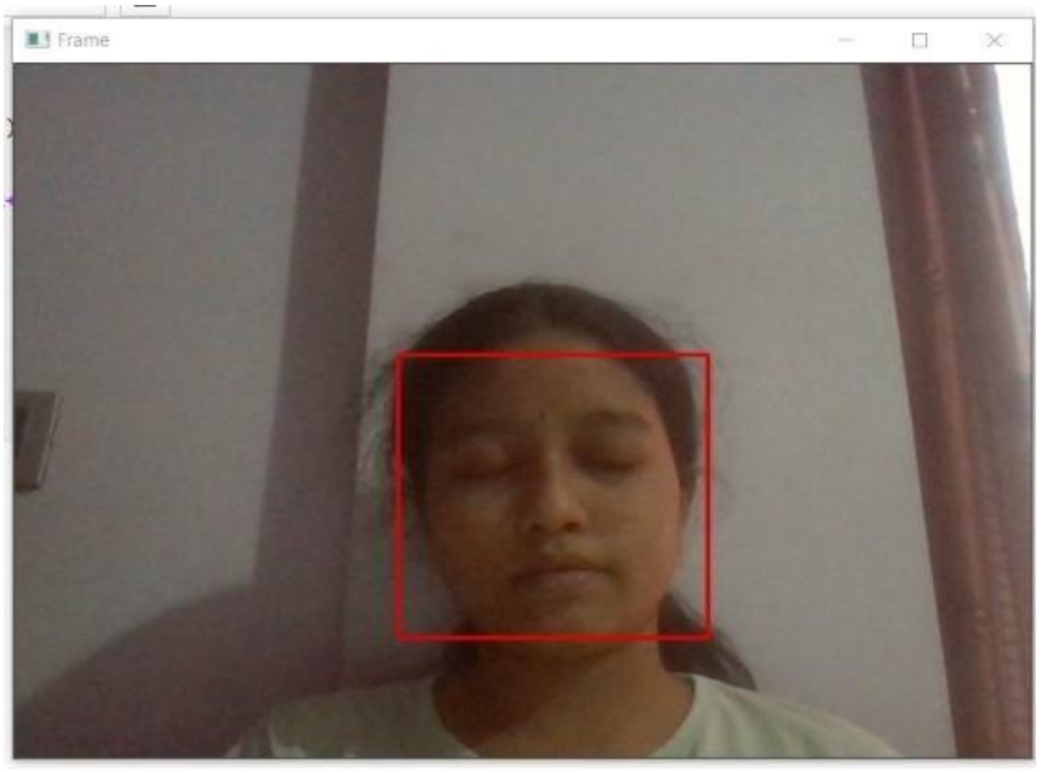


Fig. 11.5. Drowsy Output

Chapter 12

FUTURE SCOPE AND APPLICATIONS

12.1 FUTURE SCOPE

The work can be extended by combining physiological approach techniques, so the driver can be detected as drowsy through those readings. In special situations like when a driver wares spectacles along with mask then we can use the physiological reading of the driver. We can detect the drowsiness of the driver more accurately with these readings.

We can use the devices which are mentioned earlier and detect the drowsiness of the driver. While this is a research project, there is scope when this completely turns out to be developed into an application which can be run by the end users on their own for their own purposes on their own systems.

12.2 APPLICATIONS

- Wearable technology and eye detection can be integrated into vehicles as standard safety feature.
- Integrate the alarm into the car stereo system.
- Integrate armband with car ignition system such that the car won’t start if the armband is not worn.

Chapter 13

CONCLUSION

The paper described an drowsiness detection system based on CNN-based Machine Learning. We used OpenCV to detect faces and eyes using a haar cascade classifier and then we used a CNN model to predict the status. Finally, driver fatigue is evaluated. The experimental results demonstrate that when the drowsy counts reaches to 5, the driver can be considered in a fatigue state. The system was able to detect facial landmarks from images and pass it to a CNN-based trained Deep Learning model to detect drowsy driving behavior. There are limitations to this technology, such as obstructing the view of facial features by wearing sunglasses and bad lighting conditions. However, given the current state, there is still room for performance improvement and better facial feature detection even in bad lighting conditions. Thus we have successfully designed a prototype drowsiness detection system using OpenCV software and Haar Classifiers. The system so developed was successfully tested, its limitations identified and a future plane of action developed.

REFERENCES

[1] Kaushish, V., Singh, N., Maggo, K., Nagrath, P., & Jain, R. (2021). Driver drowsiness detection system with opencv and keras. SSRN Electronic Journalpp. 1-11.
https://ssrn.com/abstract=3884784

[2] Yang, J.H., Mao, Z.H., Tijerina, L., Pilutti, T., Coughlin, J.F., Feron, E. (2009). Detection of driver fatigue caused by sleep deprivation. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 39(4): 694-705. https://doi.org/10.1109/tsmca.2009.2018634

[3] Hu, S., Zheng, G. (2009). Driver drowsiness detection with eyelid related parameters by support vector machine. Expert Systems with Applications, 36(4): 7651-7658.
https://doi.org/10.1016/j.eswa.2008.09.030

[4] Mardi, Z., Ashtiani, S.N., Mikaili, M. (2011). EEG-based drowsiness detection for safe driving using chaotic features and statistical tests. Journal of Medical Signals And Sensors, 1(2): 130-137.
https://doi.org/10.4103/2228-7477.95297

[5] Noori, S.M., Mikaeili, M. (2016). Driving drowsiness detection using fusion of electroencephalography, electrooculography, and driving quality signals. J Med Signals Sens, 6: 39-46.
https://doi.org/10.4103/2228-7477.175868

[6] Dang, K., Sharma, S. (2017). Review and comparison of face detection algorithms. 7th International Conference on Cloud Computing, Data Science & Engineering, pp. 629-633.
https://doi.org/10.1109/confluence.2017.7943228

[7] VenkataRamiReddy, C., Kishore, K.K., Bhattacharyya, D., Kim, T.H. (2014). Multi-feature fusion based facial expression classification using DLBP and DCT. Int J Softw Eng Appl, 8(9): 55-68.
https://doi.org/10.14257/ijseia.2014.8.9.05

[8] Ramireddy, C.V., Kishore, K.V.K. (2013). Facial expression classification using Kernel based PCA with fused DCT and GWT features. ICCIC, Enathi, 1-6. https://doi.org/10.1109/iccic.2013.6724211

[9] Krajewski, J., Sommer, D., Trutschel, U., Edwards, D., Golz, M. (2009). Steering wheel behavior based estimation of fatigue. The Fifth International Driving Symposium on Human Factors in Driver Assessment, Training and Vehicle Design, pp. 118-124.
https://doi.org/10.17077/drivingassessment.1311

[10] Danisman, T., Bilasco, I.M., Djeraba, C., Ihaddadene, N. (2014). Drowsy driver detection system using eye blink patterns. 2010 International Conference on Machine and Web Intelligence, Algiers, Algeria, pp. 230-233. https://doi.org/10.1109/icmwi.2010.5648121

[11] Abtahi, S., Hariri, B., Shirmohammadi, S. (2011). Driver drowsiness monitoring based on yawning detection. 2011 IEEE International Instrumentation and Measurement Technology Conference, Binjiang, pp. 1-4. https://doi.org/10.1109/imtc.2011.5944101

[12] Dwivedi, K., Biswaranjan, K., Sethi, A. (2014). Drowsy driver detection using representation learning. Advance Computing Conference (IACC), IEEE, pp. 995-999.
https://doi.org/10.1109/iadcc.2014.6779459

[13] Alshaquaqi, B., Baquhaizel, A.S., Amine Ouis, M.E., Boumehed, M., Ouamri, A., Keche, M. (2013). Driver drowsiness detection system. 8th International Workshop on Systems, Signal Processing and Their Applications (WoSSPA). https://doi.org/10.1109/wosspa.2013.6602353

[14] Park, S., Pan, F., Kang, S., Yoo, C.D. (2017). Driver drowsiness detection system based on feature representation learning using various deep networks. In: Chen CS., Lu J., Ma KK. (eds) Computer Vision – ACCV 2016 Workshops, Lecture Notes in Computer Science, pp. 154-164.
https://doi.org/10.1007/978-3-319-54526-4_12

[15] Tadesse, E., Sheng, W., Liu, M. (2014). Driver drowsiness detection through HMM based dynamic modeling. 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, pp. 4003-4008. https://doi.org/10.1109/ICRA.2014.6907440

[16] Said, S., AlKork, S., Beyrouthy, T., Hassan, M., Abdellatif, O., Abdraboo, M.F. (2018). Real time eye tracking and detection- a driving assistance system. Advances in Science, Technology and Engineering Systems Journal, 3(6): 446-454. https://doi.org/10.25046/aj030653

[17] Picot, A., Charbonnier, S., Caplier, A. (2012). On-Line Detection of drowsiness using brain and visual information. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 42(3): 764-775. https://doi.org/10.1109/tsmca.2011.2164242

[18] Mandal, B., Li, L., Wang, G.S., Lin, J. (2017). Towards detection of bus driver fatigue based on robust visual analysis of eye state. IEEE Transactions on Intelligent Transportation Systems, 18(3): 545-557. https://doi.org/10.1109/tits.2016.2582900

[19] Jabbar, R., Al-Khalifa, K., Kharbeche, M., Alhajyaseen, W., Jafari, M., Jiang, S. (2018). Real-time driver drowsiness detection for android application using deep neural networks techniques. Procedia Computer Science, 130: 400-407. https://doi.org/10.1016/j.procs.2018.04.060

[20] Viola, P., Jones, M.J. (2004). Robust real-time face detection. International Journal of Computer Vision, 57(2): 137-154. https://doi.org/10.1023/b:visi.0000013087.49260.fb

[21] Jensen, O.H. (2008). Implementing the Viola-Jones face detection algorithm (Master's thesis, Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark).

[22] O'Shea, K., Nash, R. (2015). An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458.
https://arxiv.org/abs/1511.08458v2

[23] Kim, K., Hong, H., Nam, G., Park, K. (2017). A study of deep CNN-based classification of open and closed eyes using a visible light camera sensor. Sensors, 17(7): 1534.
https://doi.org/10.3390/s17071534

[24] Lee, K., Yoon, H., Song, J., Park, K. (2018). Convolutional neural network-based classification of driver's emotion during aggressive and smooth driving using multi-modal camera sensors. Sensors, 18(4): 957.
https://doi.org/10.3390/s18040957