Real Time Driver Drowsiness Detection System

A Project Report submitted in partial fulfilment of the requirements for the award of the degree of

Bachelor of Technology
in

Computer Science and Engineering

by
Ajit Sharma (191500067)
Anuj Singh (191500138)
Sushant Gautam (191500833)
Vipul Kumar (181500802)

Under the Guidance of

Dr. Rakesh Kumar
(Assistant Professor)

Department of Computer Engineering & Applications
Institute of Engineering & Technology

GLA University
Mathura- 281406, INDIA
May, 2022

Department of Computer Engineering and Applications
GLA University, 17 km Stone, NH#2, Mathura-Delhi Road, P.O. Chaumuhan, Mathura-281406 (U.P.)

Declaration

I hereby declare that the work which is being presented in the B.Tech. Project **"Driver Drowsiness real-time detection System"**, in partial fulfillment of the requirements for the award of the ***Bachelor of Technology*** in Computer Science and Engineering and submitted to the Department of Computer Engineering and Applications of GLA University, Mathura, is an authentic record of my own work carried under the supervision of **Dr. Rakesh Kumar (Assistant Professor).**

The contents of this project report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree.

Sign ______________________	Sign ______________________
Ajit Sharma 191500067	Anuj Singh 191500138
Sign ______________________	Sign ______________________
Sushant Gautam 191500833	Vipul Kumar 181500802

Certificate

This is to certify that the above statements made by the candidate are correct to the best of my/our knowledge and belief.

Supervisor
(**Dr Rakesh Kumar**)
Associate Professor
Dept. of Computer Engg, & App.

Project Co-ordinator
(Dr. Mayank Srivastava)
Associate Professor
Dept. of Computer Engg, & App.

Program Co-ordinator
(**Dr Rakesh Kumar**)
Assistant Professor
Dept. of Computer Engg, & App.

Date:

ACKNOWLEDGEMENT

We have taken efforts in this project. However, it would not have been possible without the kind support and help of our mentor. I would like to extend my sincere thanks to all of them.

We are highly indebted to Dr. Neeraj Gupta for his guidance and constant supervision as well as for providing necessary information regarding the project & also for his support in completing the project.
We would like to express my gratitude towards our faculty of GLA University & members for their kind co-operation and encouragement which help us in completion of this project.

Sign ____________________	Sign ____________________
Ajit Sharma University Roll No-181500067	Anuj Singh University Roll No-191500138
Sign ____________________	Sign ____________________
Sushant Gautam University Roll No-191500833	Vipul Kumar University Roll No-181500802

ABSTRACT

Road crashes and related accidents are a common cause of injury and death among people. According to 2015 data from the World Health Organization, road accidents have killed at least 1.25 million people worldwide, which means that every 25 seconds a person will have a fatal accident. Although the cost of road accidents in Europe is estimated at 160 billion Euros, sleep drowsiness causes about 100,000 accidents a year in the United States alone as reported by the American National Highway Traffic Safety Administration (NHTSA). In this paper, a new way of finding sleep is actually proposed. This method is based on the in-depth learning method that can be used in Android applications with high accuracy. The main contribution of this work is the pressure of the heavy-duty model on the lightweight model. In addition, the small network structure is designed based on the discovery of an important facial history point to determine if the driver is drowsy. The proposed model is capable of achieving more than 85% accuracy.

CONTENTS

Chapter 1

Introduction

1.1 Motivation and Overview

Driving activities require full attention and great brain power. In addition, driving requires good coordination of hand and foot movements, gears moving while looking at other vehicles. Lack of attention can be detrimental to drivers, pedestrians, and economic activities. Therefore, it is important so that the country has strong laws and regulations to maintain the safety of land transport. In addition, the complexity and road features can influence the driver's attention. For example, in India the current road infrastructure is far beyond the number of vehicles produced and sold is unequal in terms of road infrastructure. About 20% of people admitted that they had ever slept while driving, while 40% of people admitted that this happened at least once in their driving career. Studies show that, in India, 40% of highway accidents or nearby accidents occur as a result of drowsiness and more than 50% of all fatal road accidents involving more than two vehicles are related to alcohol. More than 65% of all single vehicle accidents are related to alcohol consumption. Looking at these statistics, it is important that we improve the driver security system. In order to develop such a system, we need to measure the condition of the driver.

This is especially true in big cities like Delhi, Mumbai and Chennai. India recorded 3,54,796 cases of road accidents in 2020 in which 1,33,201 people were killed and 3,35,201 were injured, said the NCRB, which works under the Union Home Affairs Department. Government should not always be responsible for controlling vehicle growth and poor infrastructure. In addition to these traffic elements, drivers break traffic rules (speeding, drinking, and traffic light offense) and is a major culprit in car accidents. Even enough road infrastructures and the best people driving, accidents are inevitable. According to researches, there is a strong correlation between fatigue and safety risks in driving. Fatigue may be affected by health and sleep disorders. Excessive fatigue can lead to drowsiness which has been identified as a cause of road accidents and can lead to serious injuries, as well as serious accidents of death. In this regard, drowsiness is called a decrease or loss of consciousness that has led the driver to sleep while driving.

1.2 Objective

In this context, it is important to use new technologies to design and build systems that can monitor drivers and measure their level of attention throughout the entire driving process. In this paper, an ADAS (Advanced driver assistance System) module is introduced to reduce the number of accidents caused by driver fatigue and thus improve road safety. This program manages the automatic detection of driver sleep based on visual knowledge and performance intelligence.

We propose an algorithm for detecting, tracking and analysing both the driver's face and eyes to measure PER-CLOS (percentage eye closure). PERCLOS metrics (Percentage of eyelids CLOSED) used to do so measuring sleepiness at work “Eye Tracking based driver fatigue monitoring and warning system". The system measures non-parameter values of to get drowsy the variability of the steering wheel is considered a determinant the amount of sleep because the drivers make the variation greater as the driver becomes more drowsiness. PERCLOS metropolitan alert metrics are used for detection drowsiness in heavy vehicles, alertness and warning of the driver, from the line warnings and detection of drowsiness in drivers.

1.3 Scope

This system can be further utilized and employed to alert/notify the driver during the night, which happens to be the prime time for the drivers to feel sleepy or fall asleep. A night vision camera can be employed for this purpose to deliver accurate results.

Furthermore, a video dataset or live videos could be used as input to our model which would further divide the acquired video into a stream of images for further CNN processing and classification into any one of the categories or defined classes, making our model more reliable and dependable. Video Dataset can be created by the programmer or the team members and can be later used for training purposes to provide better optimized results.

Chapter 2
Software Requirement Analysis

<u>Authors and their proposed algorithm</u>

[1] As sleep creates chances of accidents and crashes in the world, the researchers proposed different solutions ranging from finding patterns in driving habits to analysing drowsiness of the driver while driving. Most of the solution is predictive, measuring the drowsiness counting alert system through machine learning. The most common proposed method is described below: -

One of the approaches given by the author **McDonald et al.** he creates a contextual and temporal approaches algorithm that gives steering angles, vehicle speeds and accelerator pedal positions. After that these values are passed into the Bayesian Network if the driver shows signs of drowsiness. The algorithm was found to have lower false values than PERCLOS methods, which predicted drowsiness based on eyelid movements and patterns. The implication of this study is that accurate prediction, context is important. The data that you photograph in the past 10 seconds is important in understanding whether a person is at risk of taking a sleep-related route.

Many authors give solutions to drowsiness through the power of computer vision, means Using Deep Learning. It is tools that provide new tools to computer vision for detection and classification. Computer-related applications use these methods for object acquisition, health and wellness, and agricultural systems. **Tayyab Khan** proposed a solution to measure the curvature of the eyelid and thus determine whether the eyes were closed or not. They have achieved 95% accuracy in this way, but the limit is that there needs to be enough light for this method to work as it does not work well at night.

The way most companies try to follow is to combine the two different strategies described above and use multiple inputs to reach a drowsiness app decision by combining blinking vision, eye tracking, and critical monitoring Smart glass monitors these features and delivers. drowsy to intervene in a hurry and ask the driver to rest. Integration methods require multiple sensors such as infrared, cameras and heart rate monitors in one location to produce the best results.

All of these new processes in the deep learning environment come at a cost, a great model with high computational requirements. This is a feature of deep learning. In our previous work. This model that distinguishes drowsy images using a multi-layer perceptron-based model has been proposed. The result was a decent 81% accuracy when constructing a model that was 100KB in storage size.

[2].The author H. Varun Chand and J. Karthikeyan in 2021 proposed a novel model for the distribution of multi-level driver drowsiness using Convolution Neural Networks (CNN) followed by emotional analysis. Emotional analysis, in this proposed model, analyses the driver's attitude which identifies the dynamic features of various driving patterns. These driving patterns are analysed based on the acceleration system, vehicle speed, Revolutions per Minute (RPM), and driver's face recognition. The driver's face pattern is handled via 2D Convolution Neural Network (CNN) to determine the driver's behaviour and mood. Proposed model is implemented using OpenCV and the test results confirm that the proposed model detects the driver's emotions and drowsiness available technology.
The main contributions of this research paper are:

- Driver sleep detection system based on Convolution Neural Network (CNN), to detect driver, driver fatigue and vehicle acceleration system.

- The driver's mind is monitored using a novel tool developed by him, Driver Emotion Detection Classifier (DEDC), where the driver's mind is spread over many levels of rage, disgust, fear, joy, sadness and neutrality

The proposed CNN model for driver drowsiness system has two levels of flexibility order filters (n, 2n). These two levels of integrated filtration filters reduce the complexity of the model and also reduces the error detection time to 1.0 seconds leading to faster detection and notification of the driver's drowsiness.

The initial module of detecting the driver fatigue and other aforementioned status of the driver is classified using Convolution Neural Networks (CNN). The experimental result analysis was performed based on accuracy level and the error rating in detecting the driver state. As an initial analysis, the accuracy level of Convolution Neural Network (CNN) is compared with the conventional classifiers of KNN classifier and SVM classifier and its derived classifiers. Fig.1 illustrates the comparative analysis of CNN with other conventional classifiers in terms of accuracy percentage. CNN has scalable features for very large datasets and by the use of multiple convolution operations it classifies images efficiently. From Fig. 1, it is proved that the multi-layer CNN is more accurate in predicting the state of the driver and successfully classifying the multi-layer state of the driver. The accuracy level of the classifier improves by increasing the processing duration. The notable case in this driver drowsiness detection system is, it must possess a minimum to determine the distraction of the driver.

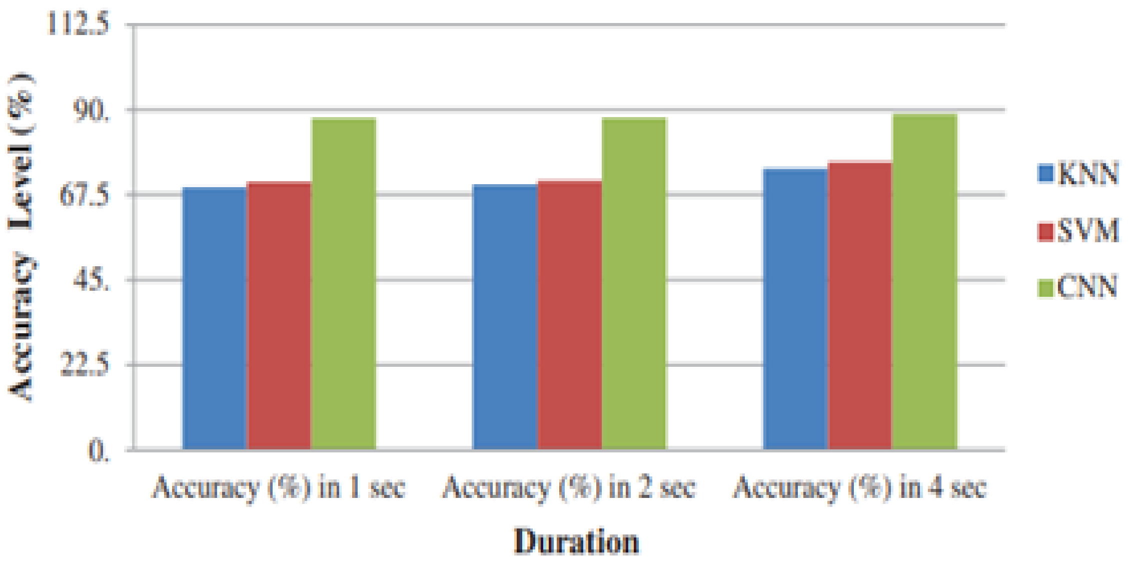


Fig.1 (Comparison of classifier types in terms of accuracy level)

[3]. Tanvi Naxane, Sayli Shrungare, Shraddha Bhandarkar, Shivani Rajhance, Mrs. Pranali Deshmukh and Tushar Kute propose a model the primary purpose of his paper was to propose a way to alert sleepy drivers in the act of driving. Most of the traditional methods to detect drowsiness are based on behavioral aspects while some are intrusive and may distract drivers, while some require expensive sensors/hardware. Therefore, in this paper, driver's drowsiness detection system is developed and implemented to aid drowsy drivers from falling asleep and to prevent accidents.

There model was based on the Behavioural Measures considering a drowsy person displays a number of characteristic facial movements and expressions, including rapid and constant blinking, nodding or swinging their head and frequent yawning. They used Neural Network Algorithms to detect person behavioural.

VGG16 as Transfer Learning:

- Due to longer training time durations, they used a shortcut to this is to use/re-use an existing model/pre-trained model and model weights that were developed for standard use.
- They used these pre-trained models combined with new model or used directly for carrying out various classification tasks.

Challenges:

- Their model was able to successfully incur satisfactory results. However, the main challenge faced was training of images in dim/low light as the model finds it difficult to process images in extremely low light which results in poor feature extraction and feature selection and therefore, wrong/inaccurate predictions.
- Another challenge that there model faced was live capturing, processing and training of videos and splitting the captured/acquired video into a series of picture frames for further processing, feature extraction and finally classification

Conclusion:

In their paper a Driver's Drowsiness Detection System proposed and using VGG16 model their model achieves an accuracy of 99.96%.

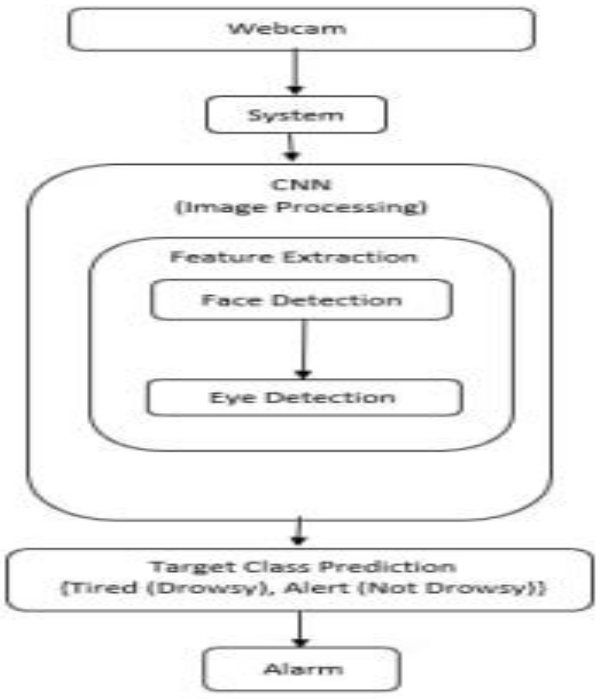


Fig 2 (system working flow graph)

NEURAL NETWORK ALGORITHMS

As we know, a neural network is made up of several interconnected computational units. Within the neural network, a basic computational unit performing specific computation or processing is called a Neuron. Neuron is mainly a node through which data computation is carried out. They receive input signals from any one the layers and it sends out the output signal to the subsequent layers.

Learning Process of Neural Networks - Learning Process of a Neural Network is divided into two phases - Forward Propagation and Back Propagation. Forward Propagation involves propagation or transferring of information from the input layer to the output layer. This whole one cycle of passing data from input layer to output layer is called Forward Propagation. Whereas, the propagation of information from output layer to the input layer is called Back Propagation. The output from the output layer is computed further to calculate the possible loss.

After we quantity or obtain loss/cost/error, our next step is to use an Optimizer. The obtained loss is analyzed and a suitable Optimizer is used to optimize the values of weights (w) and biases (b). Optimizers perform ’Back Propagation’ where the model learns. This whole process is repeated until the model delivers best optimized results.

There are mainly 3 types of layers for Neural Network based Deep Learning models - Input Layer, Hidden Layer and Output layer where the Input Layer is provided with raw input data or external data and the output from the Input Layer is fed to the Hidden Layer for further processing and finally the processed data is then given to the Output Layer which gives us the final output/prediction. Neural Networks consist many hidden layers and they can get vast in structure whereas Shallow Networks tend to have less number of layers which makes them less complex as they require more up-front knowledge of optimal features/characteristics.

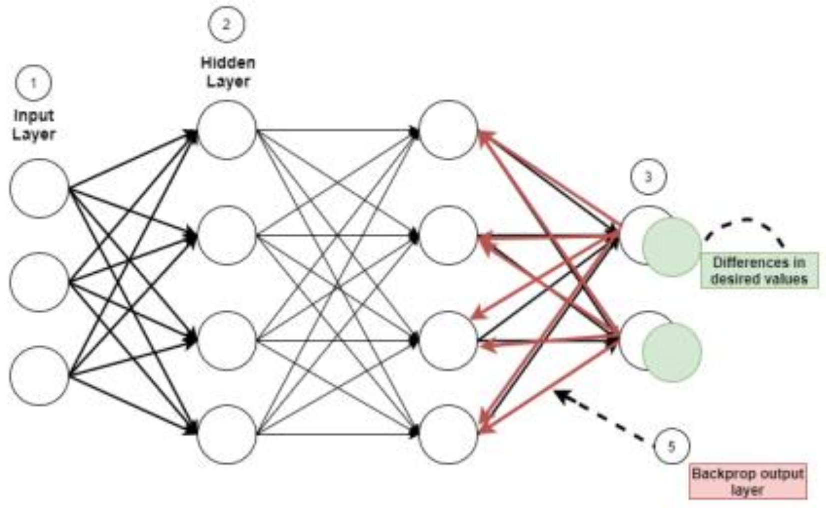
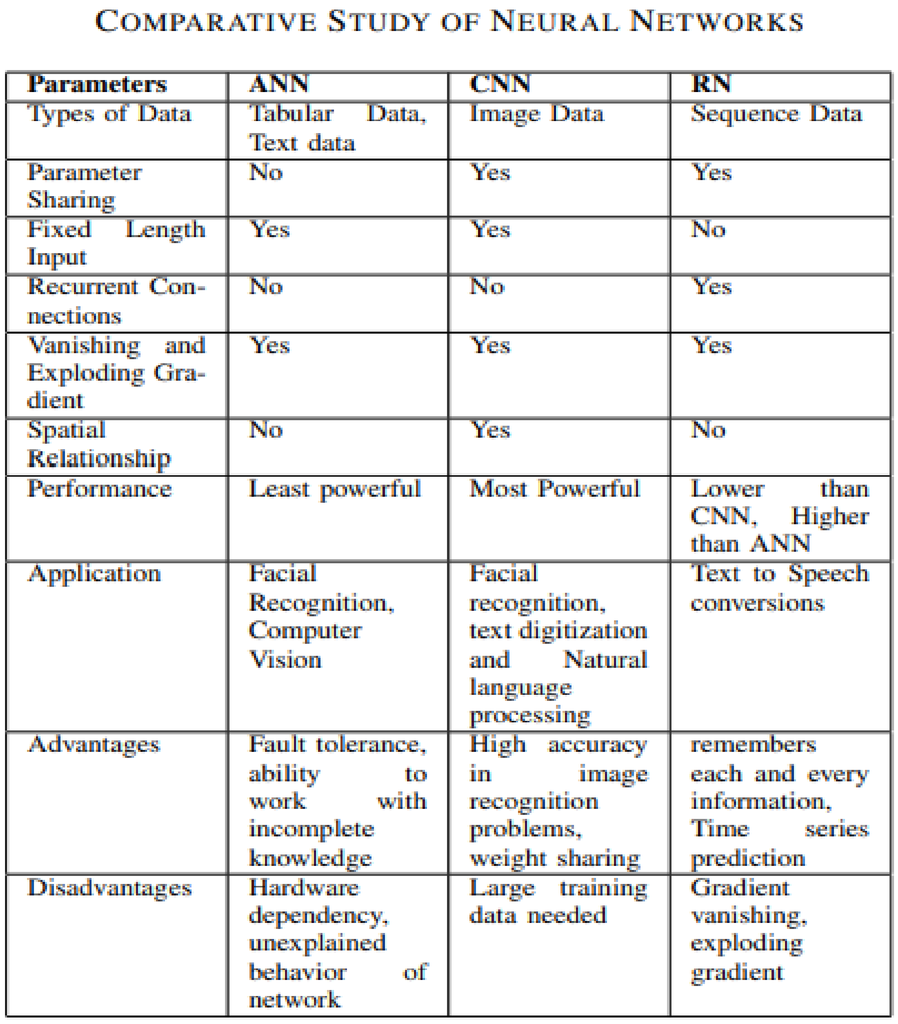


Fig 3 Learning process of neural network

There are several types of neural networks, some of the most commonly used are - Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN) and Artificial Neural Networks (ANN). For our system it was very essential to understand all the above-mentioned neural networks and get a better understanding of various neural network models



COMPARATIVE STUDY OF NEURAL NETWORKS

Parameters	**ANN**	**CNN**	**RN**
Types of Data	Tabular Data, Text data	Image Data	Sequence Data
Parameter Sharing	No	Yes	Yes
Fixed Length Input	Yes	Yes	No
Recurrent Connections	No	No	Yes
Vanishing and Exploding Gradient	Yes	Yes	Yes
Spatial Relationship	No	Yes	No
Performance	Least powerful	Most Powerful	Lower than CNN, Higher than ANN
Application	Facial Recognition, Computer Vision	Facial recognition, text digitization and Natural language processing	Text to Speech conversions
Advantages	Fault tolerance, ability to work with incomplete knowledge	High accuracy in image recognition problems, weight sharing	remembers each and every information, Time series prediction
Disadvantages	Hardware dependency, unexplained behavior of network	Large training data needed	Gradient vanishing, exploding gradient

Table 1

Since we are using images as input, we fixed CNN for our system implementation. Convolutional Neural Network helped us achieve high accuracy for our proposed model

CNN

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which takes images as input, assigns importance (learnable weights and biases) to various aspects/objects in the image and are able to differentiate/classify data in various categories/classes. 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.

CNN is majorly used for processing image data and for image classification. With the help of CNN, you can easily build a classification model. An image is nothing but a matrix with several pixel values that represent various features like brightness, shape, size, color etc. A pixel is a 1-bit number that represents either foreground or background of the image. A filter represents one particular property or a feature which is mapped to the original image to extract dominant features that helps to better generalize the image into the target output class. Filter size should be in the form of N x N, where 'N' is a positive integer representing the size of the matrix filter. The process of employing a Filter/Kernel of the size 2 x 2 or 3 x 3 to the entire input image to map/distinguish/separate out dominant or distinct features from the original image which provides us with meaningful information is called Convolution. Hence, it is named as" Convolutional Neural Network".

Convolutional Layer - Convolutional Layers are important building blocks of Convolutional Neural Networks. Convolutional Layer implements Convolutional Operation that undergoes data filtration. Kernels/Filters are applied on the actual image to extract meaningful patters or features which are later passed to the subsequent layers.

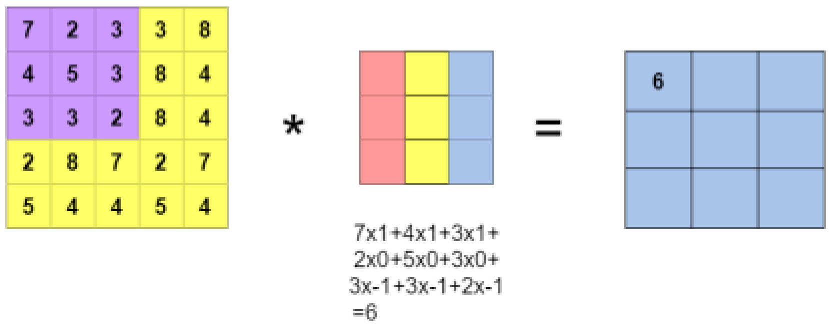


FIG 4

Fully Connected Layers- Fully Connected Layer, as the name indicates, is the layer where every neuron in one layer is connected to every neuron of another layer. Fully connected layers are used to further generalize the information obtained from the previous layers. The output of convolutional layer or a pooling layer is provided to the fully connected layer. This information in a fully connected layer is in the form of a matrix with important pixel values. This matrix is subsequently transformed into a flattened array of matrix values which is then given to the output layer to predict/display the final output.

OpenCV

OpenCV is an open-source computer source library available in python encoding to encrypt the visual skills of our smart pc. OpenCV was expected for computational capability and having a high focus on ongoing picture location and distinguishing proof. OpenCV is coded with streamlined C and can take work with multicore processors. If we need progressively programmed improvement utilizing Intel models [Intel]. These comprise of low-level schedules in different algorithmic regions which are streamlined. OpenCV therefore uses the IPP library, at runtime if that library is introduced.

Fig. 5 : This shows how the system would work and look in real world

Chapter 3
Software Design

The deep learning model developed here is trained on videos obtained from the MRL driver drowsiness detection System Dataset. These videos are pre-processed to create images for this study.

<u>Dataset and Pre-processing</u> :-

1. MRL eye dataset: - Eye detection and its components, visual acuity, and the frequency of blinking eyes are important functions in computer vision. Over the years, we have solved these tasks in the area of driver behaviour, resulting in the acquisition of a wide range of test data obtained from real-world situations. Therefore, we present the MRL Eye Dataset, a large data set of human eye images. This database contains infrared images with low and high resolution, all filmed in a variety of lightning and different devices. The database should test a few features or training separators. To make it easier to compare algorithms, the images are divided into several categories, which makes them suitable for training and testing separators

<u>Pre-processing the dataset</u>: -

Steps 1 - Extracting Photo Frames: Since we can train the neural network in video data, the images need to be extracted and embedded in the model. Video details are 30 frames per second. These frames are converted to images and are aligned with a total of about 600,000 images.

Step 2 - Data Augmentation: In the In-depth Learning Phase training phase, it is important to have additional data so that the model learns all the nuances and variations of the images. A common way to increase the number of training points is to use data augmentation. Code box was used to produce new images by creating a set of enhancement functions on images extracted from video frames.

Step 3 - Extracting landmarks from photographs: To identify and highlight important parts of the face such as eyes, mouth, nose, and jaw, using facial features. These local signals are important in measuring head position, blink detection, yawn detection, etc.

<u>Algorithm purposed</u>

Step 1 – "Take Picture as Input from a Camera"

With a webcam, we are able to take pics as input. So, to get admission to the webcam, we made a countless loop so that it will capture each frame. We use the approach supplied through OpenCV, cv2.VideoCapture (0) to get admission to the digital camera and set the seize object (cap). Cap.Read () will read each frame and we store the photograph in a frame variable.

Step 2 – "Image face detection and creation of a region of interest (ROI)"

To identify the face within the picture, we ought to begin with change over the image into grayscale as the OpenCV calculation for protest location takes gray images within the input. We don't require color data to identify the objects. We are going be utilizing the haar cascade classifier to identify faces. This line is utilized to set our classifier confront = cv2.CascadeClassifier ('way to our haar cascade xml file'). At that point we perform the location utilizing faces = face.detectMultiScale (gray). It returns a cluster of discoveries with x, y arranges, and tallness, the width of the boundary box of the protest. Presently able to emphasize over the faces and draw boundary boxes for each face.

Step 3 – “ROI detects the eyes and feeds them to the classifier”

The same strategy to identify faces is utilized to distinguish eyes. To begin with, we set the cascade classifier for eyes in l-eye and r-eye separately at that point distinguish the eyes utilizing left_eye = leye.detectMultiScale (gray). Presently we have to be extracting only the eyes information from the complete image. This may be accomplished by extricating the boundary box of the eye and after that able to drag out the eye image from the outline with code.l_eye as it were contains the picture information of the eye. This will be encouraged into our CNN classifier which is able foresee in the event that eyes are open or closed. Essentially, we'll be extricating the proper eye into r_eye.

Step 4 – “Categorizes that eyes are open or closed”

We are utilizing CNN classifier for anticipating the eye status. To bolster our picture into the demonstrate, we got to perform certain operations since the model needs the right measurements to begin with. To begin with, we change over the color picture into grayscale utilizing r_eye = cv2.cvtColor (r_eye, cv2.COLOR_BGR2GRAY). At that point, we resize the picture to 24*24 pixels as our show was prepared on 24*24 pixel pictures cv2.resize(r_eye, (24,24)). This property contains the information about two eye states (open, close). We annotated three reflection states based on the size of reflections (none, small, and big reflections).

Step 5 – “Check whether Person is Drowsy or not using PERCLOS algorithm”

The score is fundamentally an esteem we'll utilize to decide how long the individual has closed his eyes. PERCLOS, the most effective way to get drowsiness, analyses the driver's drowsiness by using visual aids. In this study, real-time eye detection under the infrared light algorithm was performed for PERCLOS calculation.

System Architecture:

When the driver is driving, the driver's face is taken by camera and is converted into video streaming. The application then analyzes video to detect drowsiness and fatigue and also checks the level of drowsiness. At this stage, the key areas to consider for analysis are: Driver's face tracking, driver's fatigue status, and recognition of important facial regions based on blindfolds and yawn. Finally, when drowsiness appears, a voice warning is given.

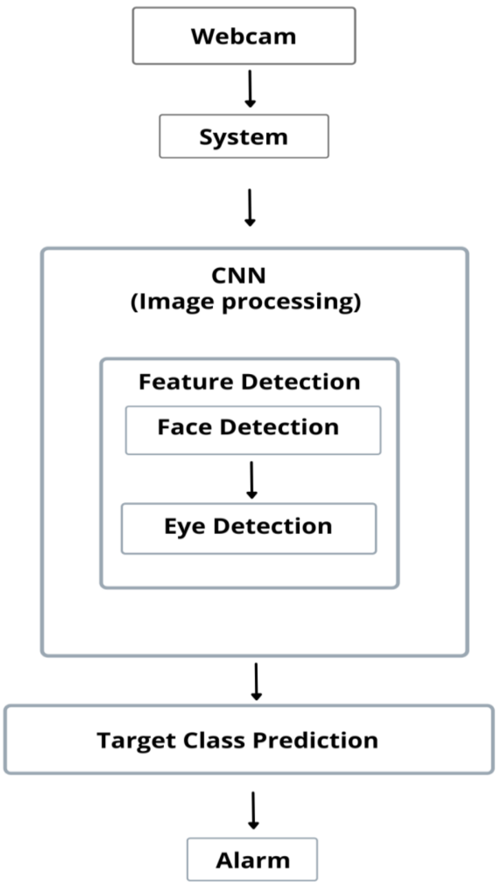
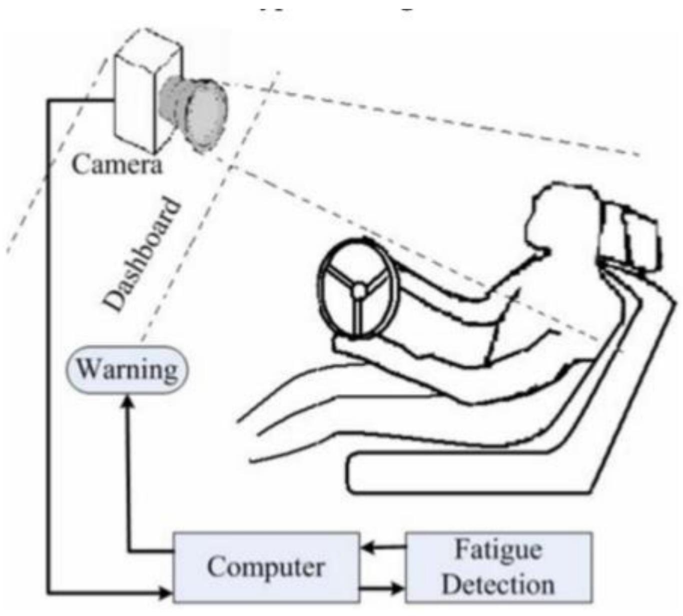


Figure 1 shows the High-Level System Architecture integration input model and pre-processing as well phase testing. Phase 1 involves pre-processing the video human face tracking stream. Phase 2 involves extraction of important facial regions such as eyes and mouth. Stage 3 including the detection of sleep symptoms such as blindness, blinking, and yawning.

A Detailed Design

The system is designed in such a way that the driver's face and mouth are constantly monitored and if the pre-defined alert levels are considered erroneous and dangerous, then the appropriate alarm is turned off and, appropriately, action is taken. to prevent any death. Figure 2 shows the Driver Sleep System Design and Yaw Detection System. It is evident that the camera is used to monitor the driver's face continuously and when it detects drowsiness or fatigue, the system on the dashboard generates a warning type warning to the driver.



(Fig.6. System Design)

Code:

Data Preparation

```
import os
import shutil
import glob
from tqdm import tqdm

Raw_DIR= r'D:\Python37\Projects\iNeuron Intership 
Projects\CV_Driver_Drowsiness_Detection\MRL Eye Data\mrlEyes_2018_01'
for dirpath, dirname, filenames in os.walk(Raw_DIR):
    for i in tqdm([f for f in filenames if f.endswith('.png')]):
        if i.split('_')[4]=='0':
            shutil.copy(src=dirpath+'/'+i, 
dst=r'D:\Python37\Projects\iNeuron Intership 
Projects\CV_Driver_Drowsiness_Detection\MRL Eye Data\Prepared_Data\Close 
Eyes')

        elif i.split('_')[4]=='1':
            shutil.copy(src=dirpath+'/'+i, 
dst=r'D:\Python37\Projects\iNeuron Intership 
Projects\CV_Driver_Drowsiness_Detection\MRL Eye Data\Prepared_Data\Open 
Eyes')
```

Model Training

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


tf.test.is_gpu_available()

batchsize=8

train_datagen= ImageDataGenerator(rescale=1./255,
rotation_range=0.2,shear_range=0.2,
    zoom_range=0.2,width_shift_range=0.2,
    height_shift_range=0.2, validation_split=0.2)

train_data=
train_datagen.flow_from_directory(r'D:\Python37\Projects\iNeuron Intership
Projects\CV_Driver_Drowsiness_Detection\MRL Eye Data\Prepared_Data\train',

target_size=(80,80),batch_size=batchsize,class_mode='categorical',subset='t
raining' )

validation_data=
train_datagen.flow_from_directory(r'D:\Python37\Projects\iNeuron Intership
Projects\CV_Driver_Drowsiness_Detection\MRL Eye Data\Prepared_Data\train',

target_size=(80,80),batch_size=batchsize,class_mode='categorical',
subset='validation')

test_datagen = ImageDataGenerator(rescale=1./255)

test_data = test_datagen.flow_from_directory(r'D:\Python37\Projects\iNeuron
Intership Projects\CV_Driver_Drowsiness_Detection\MRL Eye
Data\Prepared_Data\test',

target_size=(80,80),batch_size=batchsize,class_mode='categorical')

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

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

model.summary()
```