HARNESSING MACHINE LEARNING FOR EFFECTIVE DRIVER DROWSINESS ALERT SYSTEM

A Project Work Report submitted to the Jawaharlal Nehru Technological University Kakinada in Partial Fulfillment of the Requirements for the Award of the Degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

Submitted by

S. V. P. S. S. KAVYA
(20501A12B2)

M. MIZBA FATHIMA
(20501A1275)

S. SAHITHI
(20501A12B4)

N. RAJESWARI
(20501A1286)

Under the guidance of

Dr. G. RESHMA., M. Tech, Ph. D
Assistant Professor

Department of Information Technology

PRASAD V POTLURI SIDDHARTHA INSTITUTE OF TECHNOLOGY

(Permanent Affiliated to JNTUK: Kakinada, Approved by AICTE)

(An NBA (all UG programs) and NAAC - A+ accredited and ISO 9001:2015 certified Institution)

Kanuru, Vijayawada - 520007

April, 2024

PRASAD V POTLURI
SIDDHARTHA INSTITUTE OF TECHNOLOGY

(Affiliated to JNTU:Kakinada, Approved by AICTE)
(An ISO certified and NBA accredited institution)
Kanuru, Vijayawada – 520007

CERTIFICATE

This is to certify that the Project Work report titled "**HARNESSING MACHINE LEARNING FOR EFFECTIVE DRIVER DROWSINESS ALERT SYSTEMS"** is the bonafide work carried out by **S.V.P.S.S. KAVYA** (20501A12B2), **M. MIZBA FATHIMA** (20501A1275), **S. SAHITHI** (20501A12B4), **N. RAJESWARI** (20501A1286) in partial fulfillment of the requirements for the award of the graduate degree of **Bachelor of Technology** in **Information Technology** during the academic year **2023-2024.**

Signature of the Guide	**Signature of the HOD**
Dr. G. Reshma., M. Tech, Ph. D	**Dr. B. V. Subba Rao, M. Tech, Ph. D**

ACKNOWLEDGEMENT

First and foremost we sincerely salute our esteemed institution **PRASAD V. POTLURI SIDDHARTHA INSTITUTE OF TECHNOLOGY** for giving this golden opportunity for fulfilling our warm dreams of becoming engineers.

We hereby express our sincere gratitude to our principal **Dr. K. Sivaji Babu, M.Tech, Ph.D,** who has rendered his constant encouragement and valuable suggestions making my goals successful.

We are also thankful to our Head of Department **Dr. B. V. SubbaRao, M. Tech, Ph.D, LMISTE,MIE** for his constant encouragement and valuable support throughout the course of the project.

We are glad to express our deep sense of gratitude to **Dr. G. Reshma, M. Tech., (Ph.D).** Assistant professor, our guide, for her guidance and cooperation in completing this project successfully.

We thank one and all who have rendered help directly or indirectly in the Completion of this project successfully.

…… Project Associates

S. V. P. S. S. KAVYA	**(20501A12B2)**
M. MIZBA FATHIMA	**(20501A1275)**
S. SAHITHI	**(20501A12B4)**
N. RAJESWARI	**(20501A1286)**

ABSTRACT

Accidents often strike unexpectedly, arising from sudden circumstances beyond control. Each day, a grim tally unfolds with thousands suffering injuries or losing their lives in traffic mishaps worldwide. Alarmingly, a significant fraction of severe highway incidents stem from excessively drowsy drivers, surpassing the peril posed by intoxicated driving. In response, a paramount objective emerges: crafting a non-intrusive computer vision solution adept at swiftly identifying driver fatigue within real-time video feeds. Leveraging Haar cascades and CNN architecture integrated with Keras in Python, this innovative system aims to sound an alert upon detecting prolonged eye closure, potentially signaling drowsiness and urging immediate attention. Through continuous monitoring of facial features and eye movements, the system enhances safety by preemptively alerting drivers to impending fatigue, allowing for timely intervention and prevention of potential accidents.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1. PROBLEM DEFINITION

Driver Drowsiness has been a major factor in accidents all over the world in recent years. Many road incidents are directly caused by tired drivers. Thus, it is necessary to create systems that can identify and alert a motorist to a poor psychophysiological state, as this might greatly lower the number of incidents involving fatigued drivers. Nevertheless, there are numerous challenges in the development of these systems that pertain to accurately and quickly identifying the signs of driver weariness. Using a vision-based technique is one of the technical options for implementing driver sleepiness detection systems.

1.2. EXISTING SYSTEM

The current driver sleepiness detection system uses eye blinks to prevent accidents caused by unconsciousness. An Internet of Things (IoT)-based eye blink sensor driver drowsiness system is intended to recognize when a driver is becoming tired and notify them to stop and rest. The technology makes use of an eye blink sensor that can recognize eye blinks and a micro-controller that is designed to interpret sensor data and identify instances in which the driver is getting sleepy. When a person blinks, the electrical activity in the muscles surrounding their eyes is detected by the eye blink sensor. The sensor recognizes the changes in muscle activity and alerts the micro-controller when the driver begins to feel sleepy and blinks longer and more frequently. The driver-connected gadget in the automobile receives an alert from the micro-controller via an Internet of Things device, like a cellular modem.

1.2.1. DISADVANTAGES OF EXISTING SYSTEM

- Unreliable
- Hazardous to Retina
- Cost-Prohibitive
- Intrusive
- Not portable
- Uncomfortable or distracting for the driver.

1.3. PROPOSED SYSTEM

The Proposed System introduces an affordable, real-time driver's drowsiness detection solution, ensuring commendable accuracy. Employing a webcam-based approach, the system utilizes image processing and machine learning techniques to identify driver fatigue from facial images. OpenCV is employed for image classification, utilizing Haar cascades to pinpoint the face and eyes within the video feed. Subsequently, a deep learning model constructed with Keras assesses the driver's state, classifying it as either "drowsy" or "not drowsy." This innovative and cost-effective system integrates advanced technologies to enhance road safety by promptly identifying and alerting against potential instances of driver drowsiness.

1.3.1. ADVANTAGES OF PROPOSED SYSTEM

- Affordable
- User-friendly
- Portable
- Highly accurate performance
- Does not cause any distraction for driver

1.4. SOFTWARE AND HARDWARE REQUIREMENTS

SOFTWARE REQUIREMENTS:

- Appropriate operating system (Windows or Linux)
- Programming language environment (Python 2.7 or later versions)
- Utilization of Visual Studio Code Editor or Jupyter Notebook
- Integration of a machine learning framework (Keras)
- Inclusion of computer vision libraries (OpenCV)
- Utilization of additional libraries and tools to support the system's functionality, such as OS, Numpy, and Pygame

HARDWAREREQUIREMENTS:

- A camera (webcam) for capturing driver’s facial video.
- Personal computer
- Power supply for system operation.

1.5. MODULES DESCRIPTION

Step 1: Face and eye detection: To identify faces and eyes in a camera feed, the system employs Haar cascade classifiers. Pre-trained models that can recognize faces and eyes in a picture are called Haar cascade classifiers.

Step 2: Region of Interest (ROI) extraction: After cropping the ROI surrounding the eyes, the system utilizes it for additional processing.

Step 3: Drowsiness detection: Based on pictures of the eyes taken with a webcam, the system classifies the state of the eyes as either open or closed using a deep learning model constructed with Keras and pre-trained CNN.

Step 4: Alarm Trigger: Based on the count, the system sounds an alarm to notify the driver if the deep learning model identifies drowsiness.

1.6. SYSTEM REQUIREMENT SPECIFICATION

A System Requirements Specification (SRS) - a requirements specification for a software system - is a complete description of the behavior of a system to be developed. It includes a set of data flow diagrams that describe all the interactions the users will have with the software. Use cases are also known as functional requirements. In addition to use cases, the SRS also contains non- functional (or supplementary) requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

1.6.1 FUNCTIONAL REQUIREMENTS

In software engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, behavior, and outputs (see also software). Functional requirements may be calculations, technical details, data manipulation and processing, and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases. Generally, functional requirements are expressed in the form system shall do
<requirement>. The plan for implementing functional requirements is detailed in the system design. In requirements engineering, functional requirements specify particular results of a system. Functional requirements drive the application architecture of a system. A requirements analyst generates use cases after gathering and validating a set of functional requirements.

The hierarchy of functional requirements is: user/stakeholder request -> feature -> use case -> business rule. Functional requirements drive the application architecture of a system. A requirements analyst generates use cases after gathering and validating a set of functional requirements. Functional requirements maybe technical details, data manipulation, and other specific functionality of the project to provide the information to the user.

- Capture real-time video feeds and instantly identify faces and eyes.
- Using a pre-trained deep learning model, classify the state of the eyes.
- Monitor eye closure duration and compute sleepiness score.
- When a particular threshold is reached for drowsiness, an alert will be triggered.

1.6.2 NON-FUNCTIONAL REQUIREMENTS

A non-functional requirement in the context of requirements engineering and systems engineering is one that describes standards by which a system's performance can be evaluated rather than particular actions. The following are some of the non-functional needs for the project.

- Quick and responsive user interface. Capable of processing a lot of video streams in a short amount of time.
- Exceptionally precise in identifying and categorizing eye conditions.
- Dependable and sturdy, requiring little error or downtime.
- Simple to set up and operate, with lucid and succinct instructions.
- Excellent scalability to manage several users at once.
- Preserve user privacy by avoiding the storage of personal information.

Portability: -

The term "portability" refers to how easily the software can be deployed on all required platforms as well as the platforms that it is intended to operate on. Our product can be considered very portable as it can be operated on any operating system by employing server versions that are suitable and have been provided for multiple platforms.

Availability: -

The duration of a system's operation and usability is referred to as its "availability" or "uptime." It has to do with the server giving users the ability to view images. Our system needs to be online at all times because thousands of people will use it at any given moment. Should any updates be necessary, they must be carried out quickly and without interruption

Usability: -

Requirements for ease of use address the elements that make up the software's ability to be comprehended, learned, and utilized by its intended users. For the users, it might be simpler.

Scalability: -

The system should be scalable, able to accommodate growth as the number of users or data volume increases, and able to manage several concurrent users or video streams.

Accessibility: -

The program must adhere to applicable accessibility standards and guidelines and be made to work for all users, including those with special requirements or impairments.

Flexibility: -

It is important to prepare ahead if the organization plans to add or expand the software's capability after it has been implemented, as this will affect the decisions that are made. throughout the system's development, testing, and implementation phases. It is simple to include new modules into our system without interfering with already-existing modules or changing the applications' logical database design.

Reliability: -

There should be little downtime or crashes in the program, making it steady and dependable. It should have efficient error handling and reporting in addition to the ability to gracefully recover from mistakes and exceptions.

Performance: -

The temporal properties of the software are specified by the performance constraints.

CHAPTER 2

SYSTEM ANALYSIS

2.1 FLOW ORIENTED MODELING:

2.1.1 Data Flow Diagram (DFD):

A Data Flow Diagram (DFD) is also known as a Process Model. Process Modelling is an analysis technique used to capture the flow of inputs through a system (or group of processes) to their resulting output. The model is fairly simple in that there are only four types of symbols – process, data-flow, external entity, data store.

Process Modelling is used to visually represent what a system is doing. It is much easier to look at a picture and understand the essence than to read through verbiage describing the activities. System Analyst after talking with various users will create DFD diagrams and then show them to users to verify that their understanding is correct. The process models can be created to represent an existing system as well as a proposed system.

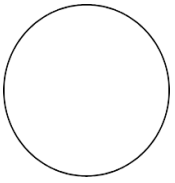
The following are some of the symbols in Process Modelling represents.

- Process
- Data Object
- Data Store
- External entity

Process:

An activity or a function that is performed for some specific reason; can be manual or Computerized; ultimately each process should perform only one activity.

- **Symbol:**



Data Flow:

Single piece of data or logical collection of information like an audio clip.

- **Symbol:**

Data Store:

Collection of data that is permanently stored.

- **Symbol:**

External Entity:

A person, organization, or system that is external to the system but intersects with it.

- **Symbol:**

The following are the levels of Data Flow Diagrams:

Level 0 DFD:

- The level 0 DFD (also known as the context level DFD) is the simplest DFD.
- The outermost level (Level 0) is concerned with how the system interacts with the outside world.
- This level basically represents the input and output of the entire system.

Level 1 DFD:

- The basic modules of the system are represented in this phase and how data moves through different module is shown.
- The level 1 DFD provides a high-level view of the system that identifies the major processes and data stores.

Level 2 DFD and other level of DFD:

- Each process from level 1 is exploded even more into processes. This decomposition continues for each level.
- The number of levels possible depends on the complexity of the system.

DFD Principles

- The general principle in Data Flow Diagramming is that a system can be decomposed into subsystems, and subsystems can be decomposed into lower-level subsystems, and so on.
- Each subsystem represents a process or activity in which data is processed. At the lowest level, processes can no longer be decomposed.
- Each 'process' (and from now on, by 'process' we mean subsystem and activity) in a DFD has the characteristics of a system.
- Just as a system must have input and output (if it is not dead), so a process must have input and output.
- Data enters the system from the environment; data flows between processes within the system; and data is produced as output from the system.

Salient Features of DFD'S

- The DFD shows flow of data, not of control loops and decisions are controlled considerations that do not appear on a DFD.
- The DFD does not indicate the time factor involved in any process whether the data flow takes place daily, weekly, monthly or yearly.
- The sequence of events is not brought out on the DFD.

Constructing a DFD:

Several rules of thumb are used in drawing DFD'S:

- Processes should be named and numbered for an easy reference. Each name should be representative of the process.
- The direction of flow is from top to bottom and from left to right. Data traditionally flows from source to destination although they may flow back to the source. One way to indicate this is to draw a long flow line back to a source. An alternative way is to repeat the source symbol as a destination. Since it is used more than once in the DFD it is marked with a short diagonal.
- The names of data stores and destinations are written in capital letters. Process and data-flow names have the first letter of each work capitalized.
- A DFD typically shows the minimum contents of a data store. Each data store should contain all the data elements that flow in and out.

Level-0 DFD:

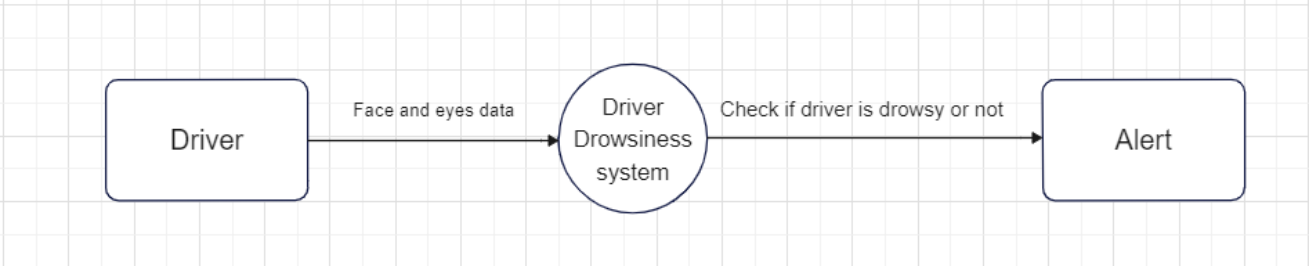


Fig 1: Level-0 diagram

Description:

Level-0, the context diagram, depicts the interaction between the driver and the drowsiness detection system. Facial and eye data from the driver are captured and analyzed bythe system using advanced algorithms to detect signs of drowsiness. This efficient process enables timely interventions, enhancing road safety by preventing accidents due to driver fatigue

Level-1 DFD:

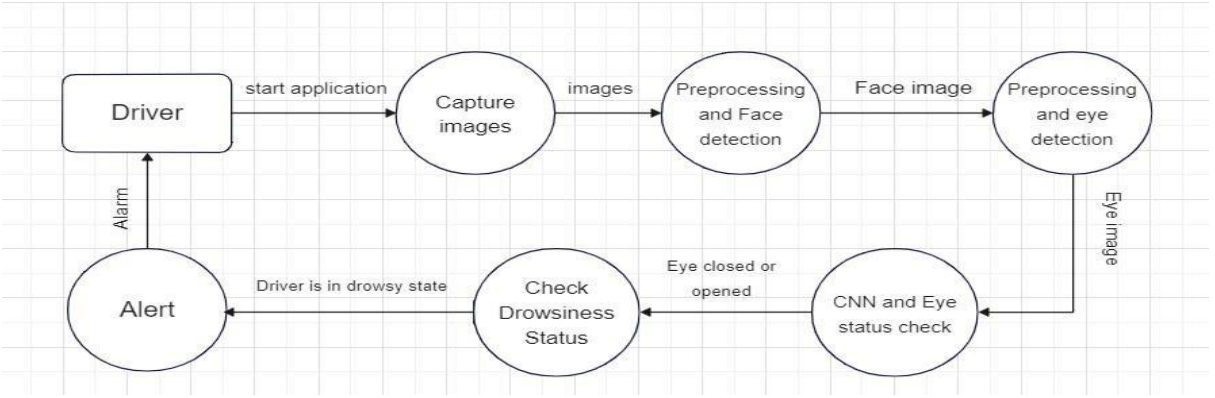


Fig 2: Level-1 Diagram

Description:

Utilizing captured images, the system employs facial and ocular recognition to identify faces and eyes. Through the implementation of convolutional neural networks (CNNs), it scrutinizes the state of the eyes, distinguishing between open and closed positions. Leveraging this information, the system makes a determination regarding the user's drowsiness. In the event of detected drowsiness, a proactive safety measure is triggered: an alarm begins to ring, serving as a timely warning to alert the user and mitigate the risk of potential accidents caused by driver fatigue.

Level-2 DFD:

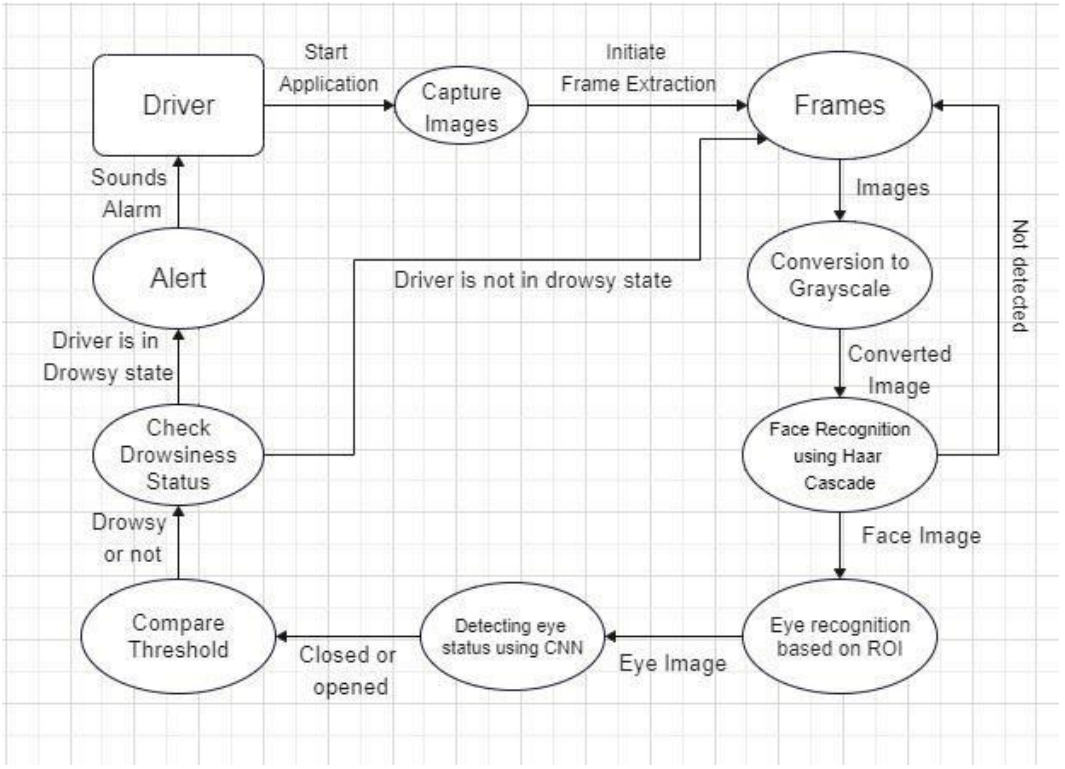


Fig 3: Level-2 Diagram

Description:

The initial step involves converting the input image to grayscale, after which the Haar cascade is employed to recognize the driver's face. Subsequently, a region of interest (ROI) is defined within the recognized face to detect the eyes. The culmination of this process entails determining driver drowsiness by comparing the CNN output with a predefined threshold value. This comprehensive approach showcases the intricate stages of image processing, facial recognition, eye detection, and neural network classification, all converging to provide an accurate assessment of the driver's alertness level.

2.1.2 Flow Chart Diagram:

A flowchart is a type of diagram that represents an algorithm or process, showing the steps as boxes of various kinds, and their order by connecting them with arrows. This diagrammatic representation illustrates a solution to a given problem. Process operations are represented in these boxes, and arrows; rather, they are implied by the sequencing of operations. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields. A flow chart diagram typically consists of shapes such as ovals, diamonds and boxes that contain words relating to a certain issue and lines with arrows connecting boxes to one another.

Flowcharts are used in designing and documenting complex processes or programs. Like other types of diagrams, they help visualize what is going on and thereby help the viewer to understand a process, and perhaps also find flaws, bottlenecks, and other less- obvious features within it. There are many different types of flowcharts, and each type has its own repertoire of boxes and notational conventions. The two most common types of boxes in a flowchart are:

- A processing step, usually called activity, and denoted as a rectangular box.
- A decision usually denoted as a diamond

Flow chart diagram:

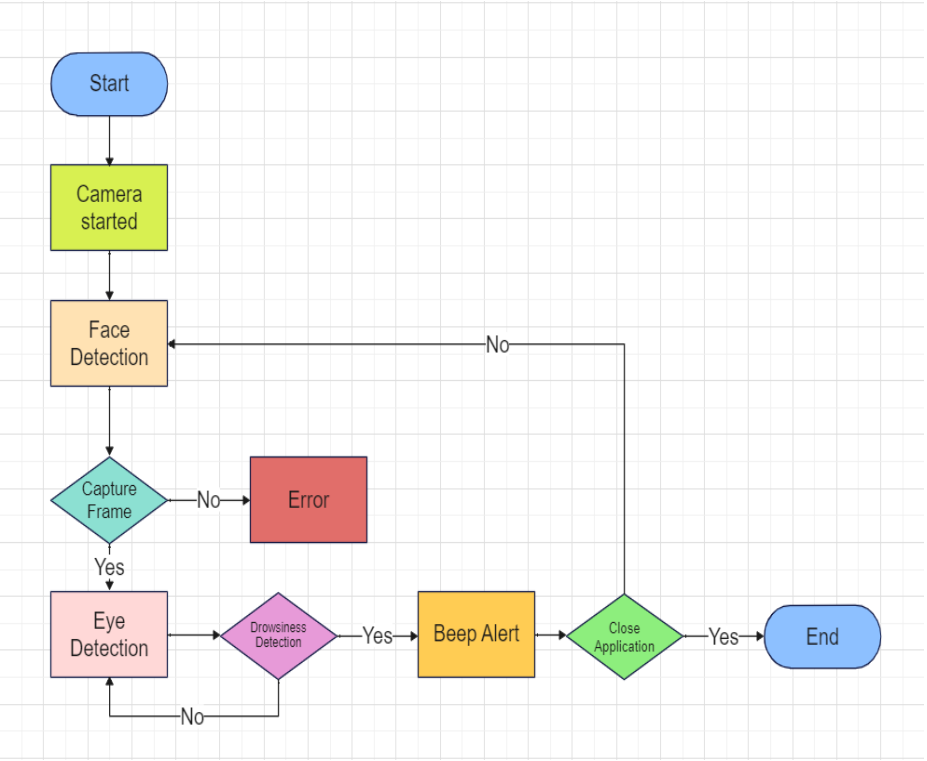


Fig 4: Flow Chart Diagram

Description:

Upon program initiation, the camera initiates to capture frames. In the event of a face being detected but a frame not being captured, an error will be raised to address the issue. Subsequently, if the system identifies the driver as drowsy, the program is terminated, and an alarm is triggered for an immediate alert. Conversely, if the driver is deemed alert, the face detection process resumes, ensuring continuous monitoring for drowsiness. This operational flow underscores the program's responsiveness to errors, its termination protocol for drowsy drivers, and its continuous vigilance for real-time face detection.

CHAPTER 3

DATA ANALYSIS

3.1 DATASET USED

The dataset for this study was produced by photographing people's eyes in various lighting scenarios with a camera. The script that was used to capture the photos divided them into two labels—"Open" and "Closed"—and put them on the local disc. Any unnecessary photos that weren't necessary for creating the model were manually removed from the dataset.

A convolutional neural network (CNN) using Keras was trained using the dataset. Multiple convolutional layers and fully connected layers made up the CNN architecture. The "models/cnnCat2.h5" file contains the final weights and model architecture after training.To categorise whether a person's eye is open or closed, which is a crucial question, the dataset was used.

Dataset:

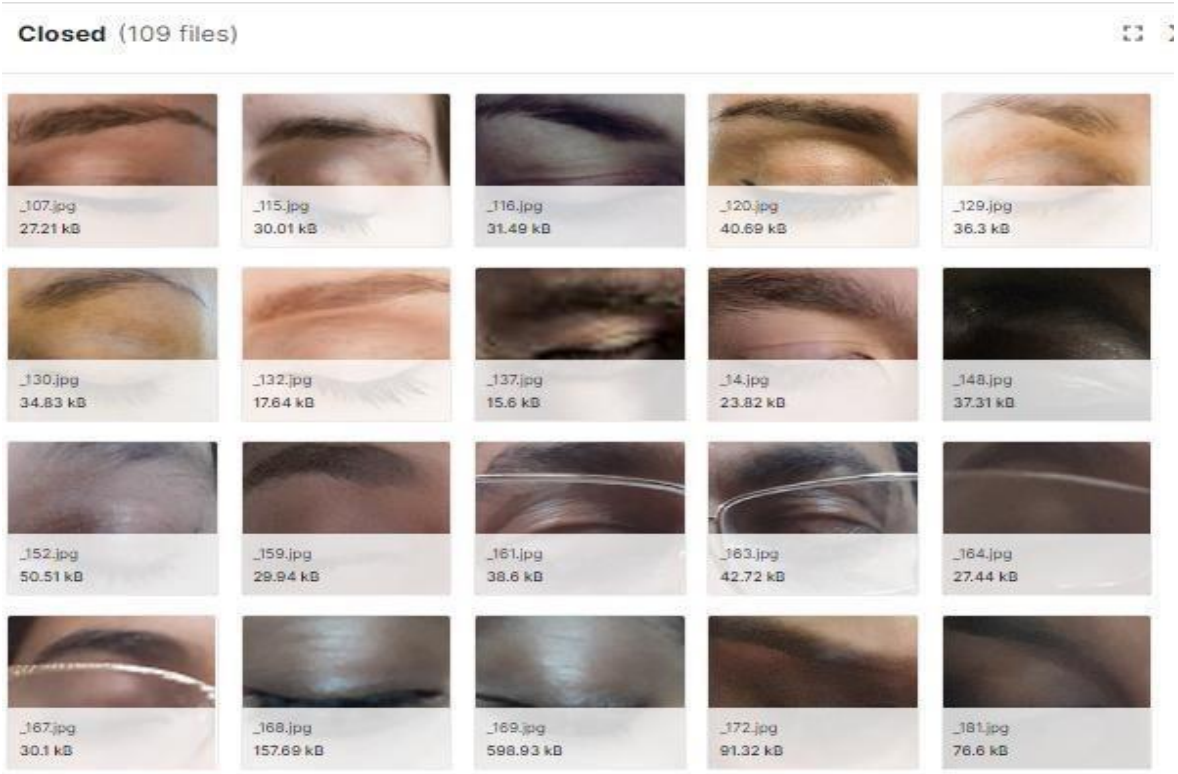


Fig: 5 - Dataset for eye closed

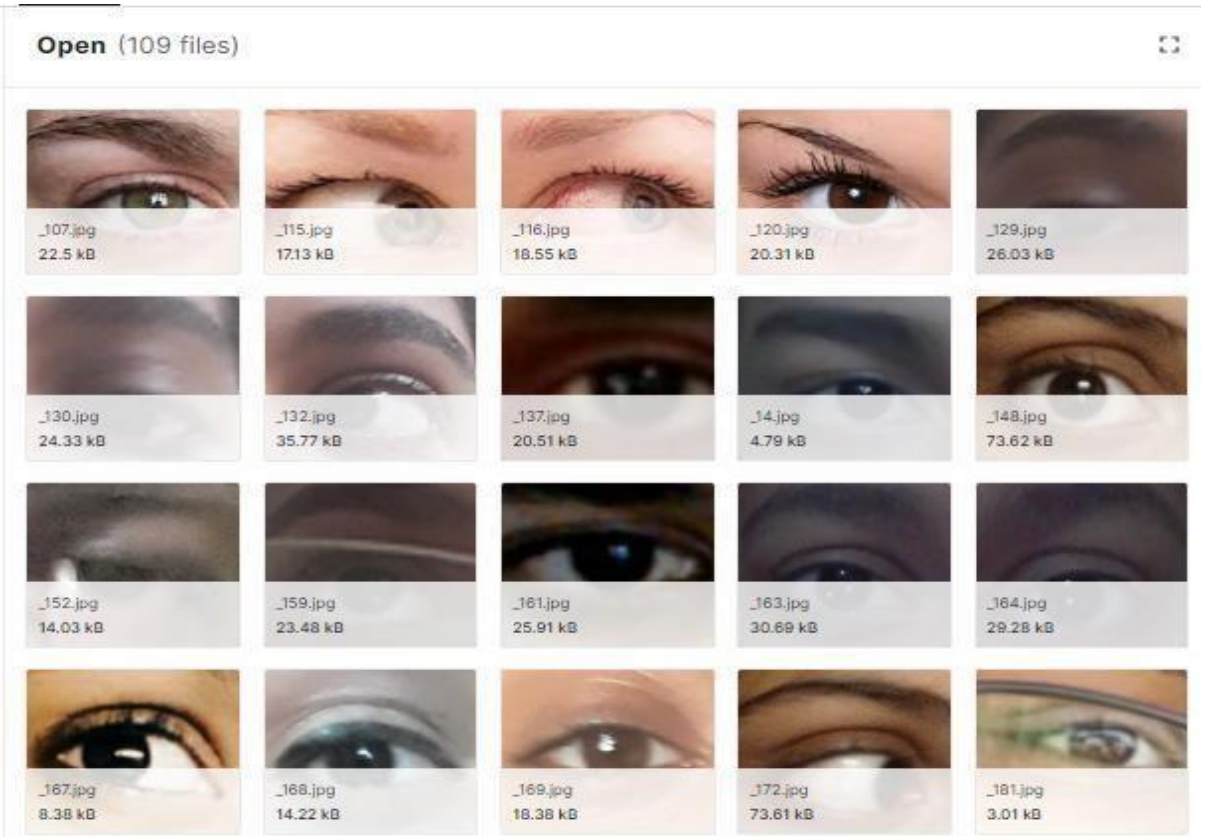


Fig: 6 - Dataset for eye opened

The Model Architecture

Convolutional neural networks (CNN), developed with Keras, were utilised to create the model that we employed. Convolutional neural networks are a specific variety of deep neural networks that excel at classifying images. In essence, a CNN is made up of three layers: an input layer, an output layer, and a hidden layer with potential for more layers. These layers are put through a convolution operation with a filter that multiplies their 2D matrices together. The CNN model architecture consists of the following layers:

- Convolutional layer; 32 nodes, kernel size 3
- Convolutional layer; 32 noes, 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. A Relu activation function is used in all the layers except the output layer in which we used Softmax.

3.2 TRAININIG OF DATASET

The programme we used to train our classification model on our dataset is contained in the "Model.py" file. This file contained a convolutional neural network implementation.

Model.py:

```
import os
from keras.preprocessing import image
import matplotlib.pyplot as plt
import numpy as np
from keras.utils.np_utils import to_categorical
import random,shutil
from keras.models import Sequential
from keras.layers import Dropout, Conv2D, Flatten, Dense, MaxPooling2D, BatchNormalization
from keras.models import load_model
def generator (dir, gen=image.ImageDataGenerator(rescale=1./255),
        shuffle=True,batch_size=1,target_size=(24,24),class_mode='categorical' ):
    return gen.flow_from_directory(dir,batch_size=batch_size,shuffle=shuffle,

color_mode='grayscale',class_mode=class_mode,target_size=target_size)
BS= 32
TS=(24,24)
train_batch = generator ('data/train',shuffle = True, batch_size = BS,target_size =TS)
valid_batch = generator ('data/valid',shuffle = True, batch_size = BS,target_size =TS)
SPE = len (train_batch.classes)//BS
VS = len (valid_batch.classes)//BS
print(SPE,VS)

model = Sequential([
    Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(24,24,1)),
```

```
    MaxPooling2D(pool_size=(1,1)),
    Conv2D(32,(3,3),activation='relu'),
    MaxPooling2D(pool_size=(1,1)),
#32 convolution filters used each of size 3x3
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(1,1)),
#64 convolution filters used each of size 3x3
#choose the best features via pooling
#randomly turn neurons on and off to improve convergence
    Dropout(0.25),
#flatten since too many dimensions, we only want a classification output
    Flatten(),
#fully connected to get all relevant data
    Dense(128, activation='relu'),
    Dropout(0.5),
#output a softmax to squash the matrix into output probabilities
    Dense(2, activation='softmax')
])
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
model.fit_generator(train_batch,
validation_data=valid_batch,epochs=15,steps_per_epoch=SPE ,validation_steps=VS)
model.save('models/cnnCat2.h5', overwrite=True)
```

The Setup Of The Model

Here is how the model is built:

1. **Importing necessary libraries/modules:**
   - os: to access file system
   - keras.preprocessing.image: to process images
   - matplotlib.pyplot: to plot images
   - numpy: for numerical computation

- keras.utils.np_utils.to_categorical: to convert class labels to one-hot encoded vectors
- random: for random number generation
- shutil: for high-level file operations
- keras.models.Sequential: to initialize a sequential model
- keras.layers: for adding layers to the model
- keras.models.load_model: to load a pre-trained model

2. Defining a generator function:

This function creates a generator object that reads images from a specified directory, applies certain augmentations/transformations (rescaling in this case), and returns batches of images and corresponding labels. The function takes parameters such as directory path, batch size, target size, etc. to customize the generator object.

3. Setting batch size and target size:

Here, the batch size is set to 32 and target size is set to (24, 24).

4. Creating generator objects for training and validation data:

Two generator objects are created using the 'generator' function defined earlier. One for training data and the other for validation data.

5. Calculating steps per epoch and validation steps:

These variables are set to calculate the number of steps (batches) required per epoch for training and validation data, respectively.

6. Building a CNN model:

A sequential model is initialized and various layers are added to it, such as convolutional layers, pooling layers, dropout layers, and dense layers.

7. Training the model:

The 'fit_generator' method is used to train the model using the training data and validate it using the validation data.

The number of epochs and steps per epoch and validation steps are specified.

8. Saving the model:

Finally, the model is saved to a file "models/cnnCat2.h5" for later use.

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 MACHINE LEARNING

Artificial Intelligence can enable the computer to think. Computer is made much more intelligent by AI. Machine learning is the subfield of AI study. Various re- searchers think that without learning, intelligence cannot be developed. There are many types of Machine Learning Techniques that are shown in fig. Supervised, Unsupervised, Semi Supervised, Reinforcement, Evolutionary Learning and Deep Learning are the types of machine learning techniques. These techniques are used to classify the data set.

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly. But, using the classic algorithms of machine learning, text is considered as a sequence of keywords; instead, an approach based on semantic analysis mimics the human ability to understand the meaning of a text.

Machine Learning is a subset of Artificial Intelligence. Machine Learning is the study of making machines more human-like in their behavior and decisions by giving them the ability to learn and develop their own programs. This is done with minimum human intervention, i.e., no explicit programming. The learning process is automated and improved based on the experiences of the machines throughout the process. Good quality data is fed to the machines, and different algorithms are used to build ML models to train the machines on

this data. The choice of algorithm depends on the type of data at hand, and the type of activity that needs to be automated.

AI manages more comprehensive issues of automating a system utilizing fields such as cognitive science, image processing, machine learning, or neural networks for computerization. On the other hand, ML influences a machine to gain and learn from the external environment. The external environment could be anything such as external storage devices, sensors, electronic segments among others.

Also, artificial intelligence enables machines and frameworks to think and do the tasks as humans do. While machine learning depends on the inputs provided or queries requested by users. The framework acts on the input by screening if it is available in the knowledge base and then provides output.

4.2 DEEP LEARNING

Deep learning (DL) is a faster expanding area of artificial intelligence that focuses on the creation of sophisticated learning and prediction algorithms and methods. Fundamentally, DL is about building sophisticated models that can efficiently learn from huge, complicated datasets and then apply that learning to make precise predictions or choices. Large-scale data-intensive issues including speech and picture recognition, natural language processing, and data analysis are particularly well suited to deep learning (DL).

Deep learning is a subfield of machine learning that is based on artificial neural networks (ANNs) inspired by the structure and function of the human brain. Deep learning algorithms are particularly good at learning from complex, high-dimensional data such as images, audio, and text. Some popular deep learning architectures include convolutional neural networks (CNNs) for image classification, recurrent neural networks (RNNs) for sequence data, and transformers for natural language processing (NLP).

The capability of DL to automatically learn representations of data at many levels of abstraction is one of its main advantages. As a result, DL algorithms are able to efficiently find patterns and features in data that could be challenging or impossible to find using other

approaches. DL models are very good at tasks like prediction, classification, and data synthesis because they can learn to recognise intricate connections and interactions between many types of information.

DL is renowned for its versatility and flexibility as well. From self-driving cars to medical diagnostics to recommendation systems, DL algorithms may be applied in a variety of settings and are easily adaptable to the unique requirements of each application.

A lot of research is being done in the DL field, and new methods and algorithms are being created all the time. As a result, DL has the potential to revolutionise many different disciplines and industries as well as open up new doors for research and innovation.

CNN

CNN (Convolutional Neural Network) is a type of deep neural network that is commonly used in image recognition and computer vision tasks. It is composed of several layers, including convolutional, pooling, and fully connected layers.

The convolutional layers are the core building blocks of a CNN, where each layer performs a set of convolutions on the input data, applying a filter or kernel to extract different features from the input image. The output of the convolutional layer is then passed through an activation function such as ReLU (Rectified Linear Unit) to introduce non-linearity into the model.

The pooling layers, which typically follow the convolutional layers, reduce the spatial size of the feature maps generated by the previous layers by downsampling the output using a max or average pooling operation. This helps to reduce the number of parameters in the model and prevent overfitting.

The fully connected layers, which are typically placed at the end of the network, are responsible for making predictions based on the extracted features. These layers take the flattened output of the previous layers as input and use a set of weights and biases to generate the final output.

CNNs are trained using a process called backpropagation, where the network's weights are adjusted to minimize a loss function that measures the difference between the predicted output and the actual output. This process is typically carried out using an optimization algorithm such as stochastic gradient descent.

CNNs have been shown to be highly effective in a wide range of applications, including image classification, object detection, and segmentation. They have also been used in natural language processing and speech recognition tasks.

Evolutionary Learning:

This biological evolution learning can be considered as a learning process: biological organisms are adapted to make progress in their survival rates and chance of having off springs. By using the idea of fitness, to check how accurate the solution is, we can use this model in a computer.

Need for Machine Learning And Deep Learning:

Machine learning and deep learning are beneficial in a variety of applications due to their capacity to learn from data, including:

- Speech and image recognition
- Translation by machine and natural language processing
- Awareness of fraud and cybersecurity
- Systems for making recommendations and customised marketing
- Quality assurance and predictive maintenance
- robots, autonomous vehicles
- planning for medical diagnosis and therapy.

Machine learning and deep learning can expedite decision-making by automating some procedures.

Challenges in Machine Learning:

There are several challenges in machine learning, including:

- **Data quality and quantity**: Machine learning algorithms require large amounts of high-quality data to learn and make accurate predictions. Obtaining and preparing such data can be time-consuming and expensive.
- **Overfitting:** Overfitting occurs when a model is too complex and fits the training data too closely, leading to poor generalization on new data. This can be mitigated by using regularization techniques and cross-validation.
- **Model selection:** There are many different machine learning algorithms to choose from, each with their own strengths and weaknesses. Selecting the best model for a particular problem can be challenging.
- **Interpretability:** Many machine learning models are considered "black boxes," making it difficult to understand how they make their predictions. This is especially important in applications where the stakes are high, such as in healthcare or finance.

- **Computational resources**: Some machine learning algorithms require significant computational resources to train, making them impractical for use on smaller devices or in real-time applications.
- **Ethical considerations**: Machine learning algorithms can perpetuate biases in data or be used to make decisions that have negative consequences for certain groups of people. It's important to consider the ethical implications of machine learning in all applications.

Applications Of Machine Learning And Deep Learning:-

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML −

- Image and speech recognition
- Natural Language Processing (NLP)
- Healthcare
- Finance
- Manufacturing and supply chain optimization
- Fraud detection and prevention
- Recommendation systems

4.3 PYTHON:

Python is a simple yet powerful programming language with excellent functionality for processing linguistic data. We chose Python because it has a shallow learning curve, its syntax and semantics are transparent, and it has good string-handling functionality. As an interpreted language, Python facilitates interactive exploration. As an object-oriented language, Python permits data and methods to be encapsulated and re-used easily. As a dynamic language, Python permits attributes to be added to objects on the fly, and permits variables to be typed dynamically, facilitating rapid development. Python comes with an extensive standard library, including components for graphical programming, numerical processing, and web connectivity. Python is heavily used in industry, scientific research, and education around the world. Python is often praised for the way it facilitates productivity, quality, and maintainability of software. NLTK defines an infrastructure that can be used to build NLP programs in Python.

It provides basic classes for representing data relevant to natural language processing; standard interfaces for performing tasks such as part-of-speech tagging, syntactic parsing, and text classification; and standard implementations for each task which can be combined to solve complex problems.

ADVANTAGES OF PYTHON:

Python has the following advantages over other languages.

1. Extensive Libraries

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

2. Extensible

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

4. Improved Productivity

The language's simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

4.4 ANACONDA NAVIGATOR/ VISUAL STUDIO

Jupyter notebook

Jupyter is a free, open-source, interactive web tool known as a computational notebook, which researchers can use to combine software code, computational output, explanatory text and multimedia resources in a single document. The Jupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.

Visual studio

Visual Studio Code is a focused code editor designed to facilitate common development tasks like debugging, task execution, and version control. It emphasizes simplicity and efficiency, aiming to offer essential tools for a swift code-edit-build-debug process, while more intricate workflows are reserved for comprehensive IDEs like Visual Studio IDE.

Numpy

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object and tools for working with these arrays.

OpenCV (cv2)

OpenCV is a popular open-source computer vision library that provides various image and video processing functions. It is often used for tasks such as object detection, face recognition, and image filtering.

Keras

Keras is a high-level neural network library that provides a simple and intuitive API for building and training deep learning models. It supports various types of neural networks, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and autoencoders.

Pygame

Pygame is a Python library for game development that provides functions for handling graphics, sound, and input. In this script, it is likely used to play audio files.

CHAPTER 5
SOURCE CODE

drowsiness detection.py

```
#Import necessary libraries
import cv2
import os
from keras.models import load_model
import numpy as np
from pygame import mixer
import time
#Initialize the mixer and load the alarm sound file
mixer.init()
sound = mixer.Sound('alarm.wav')
#Load the Haar Cascade classifiers for face detection and eye detection
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')
#Define labels and load the pre-trained CNN model
lbl=['Close','Open']
model = load_model('models/cnncat2.h5')
path = os.getcwd()
#Capture the video stream from the default camera
cap = cv2.VideoCapture(0)
#Initialize font, count, score and thickness variables
font = cv2.FONT_HERSHEY_COMPLEX_SMALL
count=0
```

```
score=0
thicc=2
rpred=[99]
lpred=[99]
#Start the video stream capture loop
while (True):
    ret, frame = cap.read()
    height,width = frame.shape[:2]
#Convert the frame to grayscale and detect faces and eyes in the frame using the Haar Cascade classifiers
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
#detects faces in the grayscale frame
    faces = face.detectMultiScale(gray,minNeighbors=5,scaleFactor=1.1,minSize=(25,25))
    left_eye = leye.detectMultiScale(gray)
    right_eye = reye.detectMultiScale(gray)
#A rectangle is drawn at the bottom of the frame
    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:
#Extract the right eye region from the frame
        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)
```

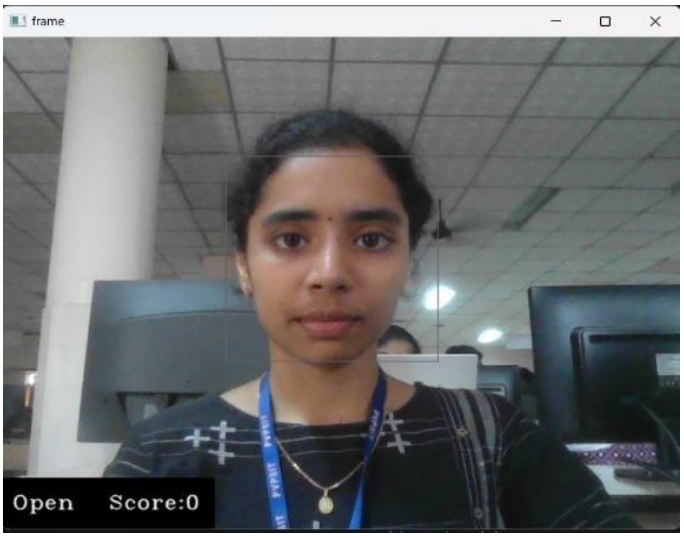
```
#check right eye is closed or opened
        if(rpred[0]==1):
            lbl='Open'
        if(rpred[0]==0):
           lbl='Closed'
        break
    for (x,y,w,h) in left_eye:
#Extract the left eye region from the frame
        l_eye=frame[y:y+h,x:x+w]
        count=count+1
        l_eye = cv2.cvtColor(l_eye,cv2.COLOR_BGR2GRAY)
        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)
#check left eye is closed or opened
        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()
```

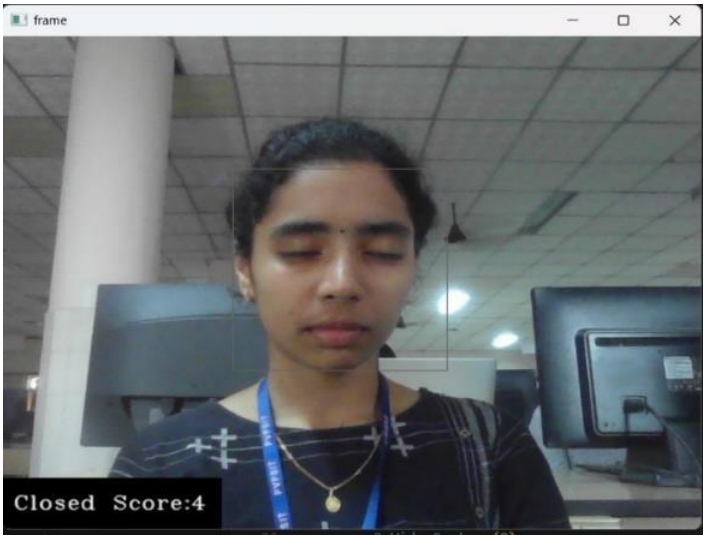
CHAPTER 6

SCREENS

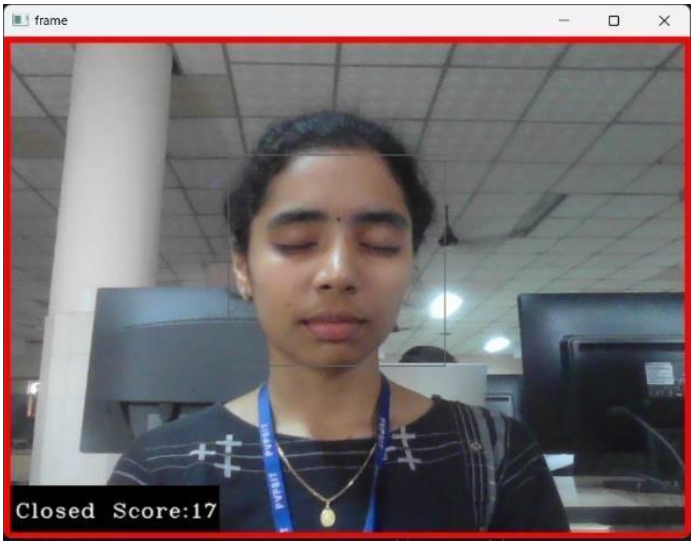
These are output screen results



Description: In the above image the eyes are labeled open and score is zero as eyes are opened.



Description: In the above image the eyes are labeled closed and score is 4 as eyes are closed. But the alarm is not ringing here as value is less than threshold.



Description: In the below image the eyes are labeled closed and score is 17 as eyes are closed. But the alarm is ringing here as value is greater than threshold.

CHAPTER 7

WORKING FUNCTIONALITY

- To identify the face in the acquired image, OpenCV would process the image.
- To concentrate on the eyes, a region of interest (ROI) would be made around the face.
- Using OpenCV, the ROI would then be further processed to find both eyes.
- Once the eyes are identified, they are put into a Deep Learning model that has already been trained to identify whether they are open or closed.
- A score would be computed based on the categorization outcome to determine if the subject is sleepy.
- The driver would be alerted by an alarm if the score went beyond a specified threshold.
- Until the user actively stops the system, it will continue to operate and repeat the steps mentioned previously.

CHAPTER 8

FUTURE SCOPE FOR FURTHER DEVELOPMENT

The Driver Drowsiness Detection System may be improved in a number of ways, including:

- **Real-time tracking**: Current system limitations only allow for analyzing camera feed images. A more advanced approach could incorporate real-time tracking to monitor driver behavior, such as looking around or engaging in other tasks, providing deeper insights into their state of alertness.
- **Integration with other sensors**: To enhance the accuracy of drowsiness prediction, integrating the system with additional sensors like heartbeat monitors and steering sensors could offer a more comprehensive understanding of the driver's condition.
- **Enhanced precision**: Implementing advanced techniques beyond simple classifiers for distinguishing between open and closed eyes could significantly improve result accuracy, leading to more reliable drowsiness detection.
- **Personalization**: Customizing the system based on individual factors such as age, gender, and medical conditions allows for tailored alerts and interventions, optimizing effectiveness for each driver.
- **Mobile application**: Developing a companion mobile application that interfaces with the system provides drivers with real-time advice on when to take breaks and insights into their fatigue levels, promoting proactive measures to combat drowsy driving.
- **Integration with vehicles:** Enabling the system to interact directly with vehicle controls, such as automatically slowing down or stopping the car if the driver becomes excessively drowsy, enhances safety measures and prevents potential accidents.

CHAPTER 9

CONCLUSION

In conclusion, the implementation of a driver drowsiness detection system integrating Haar cascade classifiers, OpenCV, Pygame, Keras, and Convolutional Neural Networks (CNN) offers a promising solution to mitigate the risks associated with drowsy driving. By leveraging computer vision techniques, the system effectively detects facial features and monitors eye movements in real-time, crucial for identifying signs of driver fatigue. The utilization of CNNs enhances the accuracy of drowsiness classification, enabling timely alerts when the driver's eyes exhibit prolonged closure, indicative of sleepiness.

With the integration of Pygame, the system delivers audible alerts in the form of alarms, effectively notifying the driver when drowsiness is detected. This proactive approach serves to prompt immediate corrective action, potentially preventing accidents resulting from drowsy driving. Furthermore, the modular design and incorporation of widely-used libraries like OpenCV and Keras ensure flexibility and scalability, facilitating easy deployment across different platforms and environments.

The accuracy of the model can be increased, it can be expanded to identify additional indicators of driver weariness, and it can be combined with additional technologies like GPS to notify the driver to take a break. This research has a lot of potential for further development.

In essence, the driver drowsiness detection system represents a critical advancement in automotive safety technology, offering a reliable means to enhance driver alertness and reduce the likelihood of drowsy driving-related incidents. Through continued research and refinement, such systems have the potential to significantly contribute to road safety and save lives by addressing the dangers posed by driver fatigue.

CHAPTER 10

REFERENCES

[1] D. MOHAN, “Road Accidents in India,” IATSS Res., vol. 33, no. 1, pp. 75–79, 2014.

[2] S. Mehta, S. Dadhich, S. Gumber, and A. J. Bhatt, “Real-Time Driver Drowsiness Detection System Using Eye Aspect Ratio and Eye Closure Ratio,” pp. 1333–1339, 2019.

[3] Wanghua Deng, Ruoxue Wu. Real-Time Driver-Drowsiness Detection System Using Facial Features. IEEE Access PP (99):1-1.

[4] Chen, Y.W., Kubo, K.: A robust eye detection and tracking technique using Gabor filters. In: Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing, IEEE, vol. 1, pp. 109–112 (2007).

[5] Francois Chollet, Deep Learning with Python, Second Edition

[6] Sumit Saha, A Comprehensive Guide to Convolutional Neural Networks — the ELI5 Way

[7] V.E. Machaca, J. P. Cahuana Nina, and K.M. Fernandez Fabian. A Survey on Drowsiness Detection Techniques. Vol 2747, Paper14

[8] Rizul Sharma, Pratyush Agarwal. Driver drowsiness detection system - ResearchGate

[9] M. Abadi et al., TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems

[10] DataFlair, AI with Python - Computer Vision Tutorials

APPENDIX- A

HARNESSING MACHINE LEARNING FOR EFFECTIVE DRIVER DROWSINESS ALERT SYSTEM

Harnessing Machine Learning for Effective Driver Drowsiness Alert System

Dr. G. Reshma[1], *Kavya Suggula[2], Fathima Mirza[3], Sahithi Surapaneni[4], Rajeswari Nidamarthi[5]
[1 2,3,4,5] Prasad V Potluri Siddhartha Institute of Technology, Kanuru, Vijayawada - 520007
[1]greshma@pvpsiddhartha.ac.in
[2*]kavyasuggula@gmail.com
[3]affumirza110@gmail.com
[4]sahithisurapaneni8779@gmail.com
[5]nidamarthi.raji@gmail.com

Abstract - Accidents often strike unexpectedly, arising from sudden circumstances beyond control. Each day, a grim tally unfolds with thousands suffering injuries or losing their lives in traffic mishaps worldwide. Alarmingly, a significant fraction of severe highway incidents stem from excessively drowsy drivers, surpassing the peril posed by intoxicated driving. In response, a paramount objective emerges: crafting a non-intrusive computer vision solution adept at swiftly identifying driver fatigue within real-time video feeds. Leveraging Haar cascades and CNN architecture integrated with Keras in Python, this innovative system aims to sound an alert upon detecting prolonged eye closure, potentially signaling drowsiness and urging immediate attention. Through continuous monitoring of facial features and eye movements, the system enhances safety by preemptively alerting drivers to impending fatigue, allowing for timely intervention and prevention of potential accidents.

Keywords: Face and Eye Detection, Python, Feature extraction, OpenCV, Keras, Pygame

1. Introduction

In our modern, fast-paced society, many individuals sacrifice adequate rest and sleep in order to fulfill the requirements of their busy lifestyles, often leading to drowsiness behind the wheel and contributing significantly to road accidents. These incidents result in approximately 1200 fatalities and 76,000 injuries each year [1].

Fortunately, advancements in technology offer promising solutions through the integration of computer vision and machine learning. These sophisticated systems utilize cameras and real-time recording to monitor driver behavior, including eyelid movements and head position, effectively assessing levels of drowsiness [2][3]. Convolutional neural networks (CNNs) are central to this process, analyzing facial landmarks and detecting eye movements to identify subtle indications of fatigue with high accuracy [4]. Through extensive training on extensive datasets, these technologies are able to consistently recognize patterns linked to fatigue. and issue timely warnings to drivers.

However, the implementation of effective drowsiness detection and alert systems presents several challenges. Real-time processing requirements, environmental variability, and individual differences in drowsiness patterns pose significant obstacles to system design and deployment. To overcome these obstacles, interdisciplinary cooperation between engineers, psychologists, and medical professionals is essential in refining drowsiness detection techniques [7].

Drowsiness can result from several circumstances, including depression, grief, irregular work schedules, stress, and travel across time zones. To mitigate accidents resulting from drowsiness, the evolution of a comprehensive Driver Fatigue Indicator System becomes imperative [8]. Rooted in engineering and computer science, this system executes as a testament to the invaluable contributions of these disciplines to society. By alerting drivers when they exhibit signs of drowsiness, this model plays a vital part in accident prevention.

Indeed, beyond its direct influence on traffic safety, the evolution and widespread adoption of Drowsiness Detection Systems hold broader societal implications. By reducing the incidence of drowsy driving-related accidents, these mechanisms support to the alleviation of burdens on healthcare systems and emergency services [14]. The prevention of injuries and fatalities translates to fewer hospitalizations, reduced rehabilitation costs, and diminished emotional trauma for individuals and families affected by accidents.

Moreover, the application of such technologies fosters a culture of responsibility and awareness about the risks associated with sleepy driving. Through public education campaigns and policy initiatives, supported by the deployment of these advanced systems, society can cultivate a collective comprehension of the importance of prioritizing rest and sleep, especially before engaging in activities like driving [12].

Furthermore, the integration of Driver Drowsiness Detection and Alarm Systems into vehicles can incentivize manufacturers to prioritize safety features in their designs, potentially leading to broader improvements in automotive safety standards. This

integration may also pave the way for advancements in autonomous vehicle technology, as similar methods for computer vision and machine learning are utilized for driver monitoring in self-driving cars.

However, the road to widespread adoption of various techniques for detecting sleepiness is not without its challenges. Public awareness campaigns, regulatory frameworks, and industry standards must align to guarantee that these technologies reach the people who require them most. Additionally, concerns regarding privacy, data security, and user acceptance must be addressed to garnertrust and confidence in these systems among drivers and passengers alike.

In summary, the fusion of OpenCV key methods for machine learning represents a beacon of hope in the ongoing battle against drowsy driving [10]. By harnessing the power of computer vision and deep learning algorithms, we have the capacity to save thousands of lives and avoid countless injuries on our roads each year. Through collaborative efforts across disciplines and industries, we can overcome the technical, regulatory, and societal challenges that stand in the way of safer highways for all. With innovation, determination, and a shared commitment to road safety, we can pave the way towards a future where Driving while unconscious is no longer a leading cause of accidents, but a relic of the past.

2. Literature review

Earlier investigations have focused on mitigating how many accidents occur on the roads attributable to driver drowsiness by developing and implementing real-time drowsiness detection and monitoring systems. P. Davidson et al. devised a straightforward system utilizing the Haar Algorithmic program within OpenCV libraries to detect facial features, particularly the eye region, by capturing images and measuring factors to determine eyelid closure extent.

Similarly, A. Paola [16] suggested a procedure employing an infrared camera to identify indicators of a driver sleepiness by tracking bright features indicative of eye movements. When the system senses drowsiness, issues a alerting the driver.

C. Kumar [17] employed the Otsu thresholding method to identify facial characteristics and utilized morphological operations and K-means clustering for accurate eye detection. Subsequently, a set of features was calculated and employed in training a non-linear Support Vector Machine (SVM) to determine eye status.

Numerous elements, such as the surrounding circumstances, physiological and biological factors, social and economic factors, and vehicle-related issues, significantly influence driver behavior and performance. Investigations have looked into both objective and subjective measurements to assess driver drowsiness, with subjective measurements often relying on pre- orpost-driving questionnaires, albeit with limitations in addressing drowsiness during driving tasks.

Over the past two decades [20], methods such as muscle fatigue estimation through SEMG (Surface Electromyography) have gained prominence in physically detecting driver fatigue. Modifications to electromyogram methods have focused on muscle metabolic processes to detect driver fatigue across various vehicle types [19].

Highly sensitive sensors, like load cell-based Nanostructured compound sensors integrated into the steering wheel, offer real-time monitoring of pressure exerted by the driver's hand, complementing facial feature-based detection systems. An integrated system combining camera-based and sensor-based solutions calculates threshold values for incoming signals to effectively monitor both physical and facial indicators of drowsiness.

Physiological signals, including Electrocardiogram (ECG), Electromyogram (EMG), Electroencephalogram (EEG), and Electrooculogram (EOG), have been utilized in previous research to detect drowsiness. EOG signals, in particular, have been employed to track changes in eye movement indicative of drowsiness [18]. Previous studies have used electrodes to measure these signals and make decisions based on parameters such as eye movement speed and pattern to alert the user of impending drowsiness.

3. Proposed system

3.1 Design Overview

Our proposed design draws inspiration from previous studies, integrating Convolutional Neural Networks (CNN) [6] with Keras [5] and OpenCV [10], while leveraging the Haar cascade classifier for real-time detection of driver fatigue. OpenCV is instrumental in image processing, whereas the Haar cascade classifier is employed to identify facial features [11]. Specifically, the CNN model is trained to discern differences in eye openness. This amalgamation of techniques seeks to improve sleepiness detection accuracy and responsiveness in real-time scenarios.

3.2 Methodology

The methodology of our approach encompasses three primary components: face and eye recognition utilizing the Haar cascade classifier, extraction of facial features employing OpenCV, and drowsiness classification through a CNN implemented in Keras. Operational in real-time, our system records visual media from a camera positioned either on the dashboard or within the vehicle.

We employed Keras to construct our model, with TensorFlow serving as the backend for Keras. Additionally, Pygame is applied to manage and play alarm sounds [13].

The OpenCV library is pivotal for detection of faces and eyes, initially processing live video streams using the Haar cascade classifier to pinpoint the driver's eyes and face. This foundational step lays the laying the foundation for further feature extraction and drowsiness classification processes.

3.3 Dataset

A Convolutional Neural Network (CNN) model is subsequently trained on a comprehensive dataset comprising images depicting eyes both open and closed, enabling the assessment of the driver's fatigue level [1]. This dataset is meticulously curated and annotated, providing labels denoting the status of eye in each image. During the procedure of training the model, it progressively learns to discern patterns and features feature of both closed and open eyes, thereby enhancing its accuracy over successive iterations.

The dataset encompasses distinct subsets utilized in training and validation, with each subset containing images depicting both open and closed eyes. This diversified dataset guarantees that the model is exposed to a diverse range of real-world scenarios, facilitating robust drowsiness detection capabilities. Upon training completion, the model computes scores indicative of drowsiness levels, enabling the triggering of alerts when necessary.

By including a variety of eye images, the model can effectively identify drowsiness in real-world scenarios, triggering alerts based on computed scores.

Fig. 1. Closed eye image

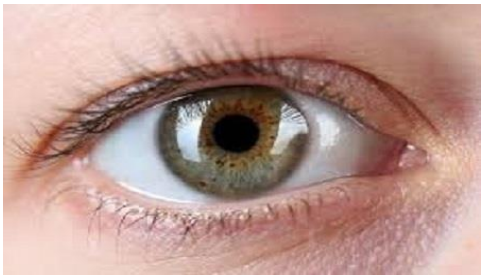


Fig. 2. Opened eye image

3.4 Model

The Convolutional Neural Network (CNN) model constructed with Keras is a fundamental component of the methodology, especially in tasks related to image classification, such as driver drowsiness detection[15].

CNNs are particularly effective in processing and analyzing visual information because of their capacity to capture spatial hierarchies offeatures within images.

The model we utilized was constructed with Keras, employing Convolutional Neural Networks (CNNs). CNNs stand out as particularly effective for image classification tasks[6]. They typically include an input layer, an output layer, and one or more hidden layers such as Convolutional layers, pooling layers, flattening layers and Dense layers also known as Fully Connected layers. Within these layers, convolutions are executed using filters, conducting 2D matrix multiplications between the layers and filters.

Activation functions are used which introduces non-linearity into the network, allowing it to understand complexity in patterns and relationships in the data.

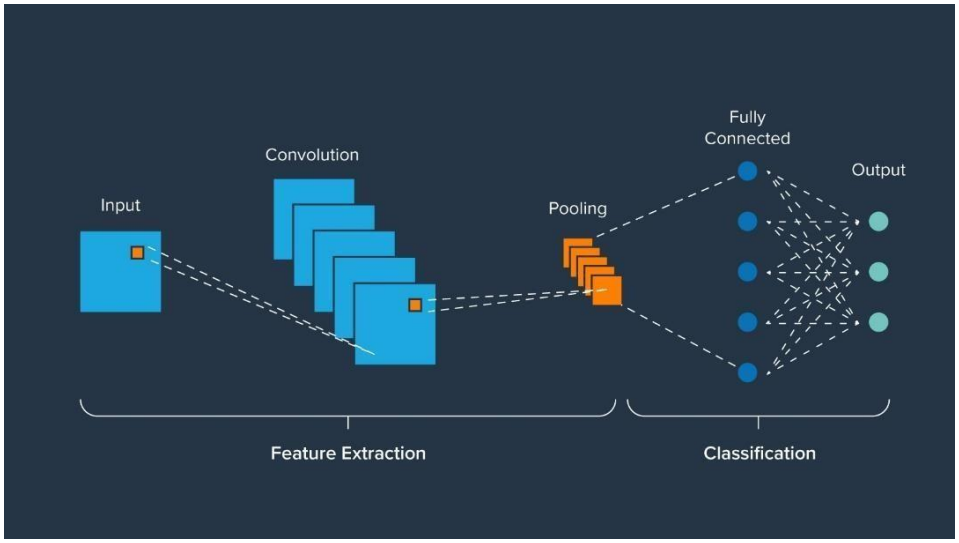


Fig. 3. Convolutional Neural Network Architecture

A Convolutional Layer in a Convolutional Neural Network (CNN) applies filters to input images to extract features[6][15]. The conv2d functionality is a key part of this layer, involving sliding filters over the input image, computing dot products, and producing feature maps. This operation utilizes concepts like filters, feature maps, stride, padding, and activation functions.It's fundamental for applications such as image recognition and object detection in computer vision.

Dimension of image = (n, n)

Dimension of filter = (f, f)

Dimension of output will be ((n – f + 1), (n – f + 1))

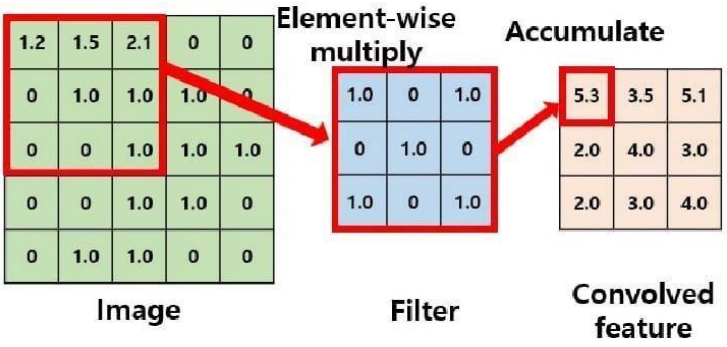


Fig. 4. Conv2D Operation

Pooling layers in CNNs reduce computing complexity by using down sample feature maps and controlling overfitting[6]. Max pooling, a popular technique, takes the maximum value out of specific input regions, maintaining significant elements while decreasing spatial dimensions[15]. This operation introduces translation invariance, making the network less sensitive to small shifts in the input data. Overall, pooling layers play an important role in extracting robust characteristics from images while improving efficiency in convolutional neural networks.

Dimension of image = (n, n)

Dimension of filter = (f, f)

Size of Stride = s

Dimension of output will be [((n - f) / s) + 1), ((n - f) / s) + 1)]

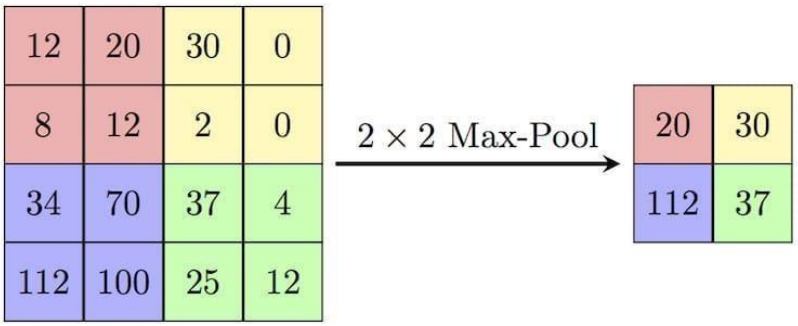


Fig. 5. Max Pooling Operation

All neurons are linked by a fully connected layer from the previous layer to each neuron in the current layer[6]. It computes a weighted sum of inputs, adds a bias term, and applies an activation function for each neuron. This process transforms the input data into a higher-dimensional representation, enabling complex nonlinear mappings. Fully connected layers are essential in deep learning models for tasks like classification and regression.

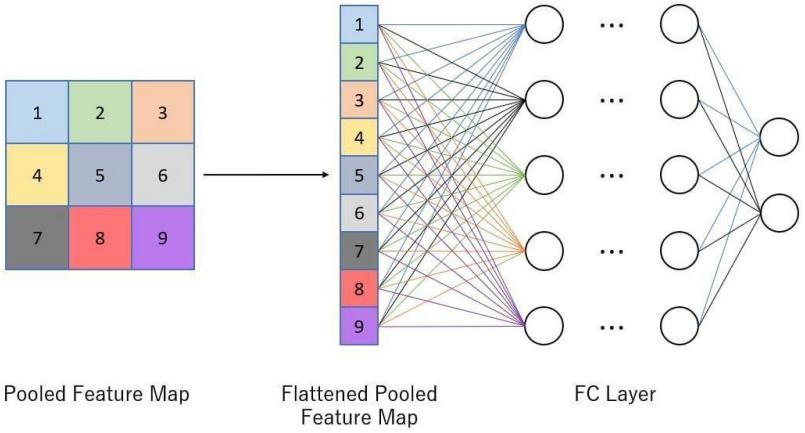


Fig. 6. Flatten and Dense Layers

The architecture of our CNN model entails:

- 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, also fully connected, consists of 2 nodes.

Throughout the layers, ReLU activation functions are employed, with Softmax utilized in the output layer. This configuration enables effective drowsiness classification, enhancing the system's overall performance and accuracy.

3.5 Algorithm

The below steps collectively form our thorough process for identifying driver fatigue in real-time.

Step 1: Take image input

- To initiate the procedure for detecting drowsiness, the system accesses the webcam, continuously capturing frames using the cv2.VideoCapture(0) method from the OpenCV library. This sets up an infinite loop, ensuring that each frame of the live footage feed is available for analysis.

Step 2: Create a Region of Interest (ROI) after Finding a Face in the Image

- Utilizing the Haar cascade classifier, the system identifies faces within each captured frame. Upon detection, the detected faces are surrounded by bounding boxes, creating regions of interest (ROI). This step is crucial for focusing subsequent analysis specifically on the areas containing the driver's face.

Step 3: Recognize the eyes in the ROI and input them into the classifier.

- With the face regions identified, the system proceeds to detect eyes within ROI. Employing cascade classifiers specifically designed for eye detection, the system isolates and extracts eye images from the frame. This procedure guarantees that only the relevant eye regions are analyzed, enhancing the accuracy of subsequent classification.

Step 4: The classifier will classify the state of the eyes as open or closed.

- The system employes a Convolutional Neural Network (CNN) classifier trained to classify eye status as either closed or open. The extracted images of eye are resized and normalized prior to being into the classifier. Leveraging learned patterns and features, the CNN predicts the status of each eye according to the provided images.

Step 5: A score is maintained to verify if the driver appears sleepy or not.

- To evaluate the driver's drowsiness level, the algorithm keeps track of a score that dynamically adjusts according to the classification results. When both eyes are classified as closed, the updated score is incremented, indicating a potential state of drowsiness. Conversely, when both eyes are classified as open, the updated score is decremented, signifying alertness.

Step 6: An alarm starts beeping until the driver opens his eyes

- If the accumulated score surpasses a predetermined cutoff point, suggesting a high likelihood of fatigue, the system sounds an alarm to notify and warn the driver. This alarm persists until the driver opens their eyes, prompting immediate attention and action to reduce the harm of drowsy driving-related incidents.

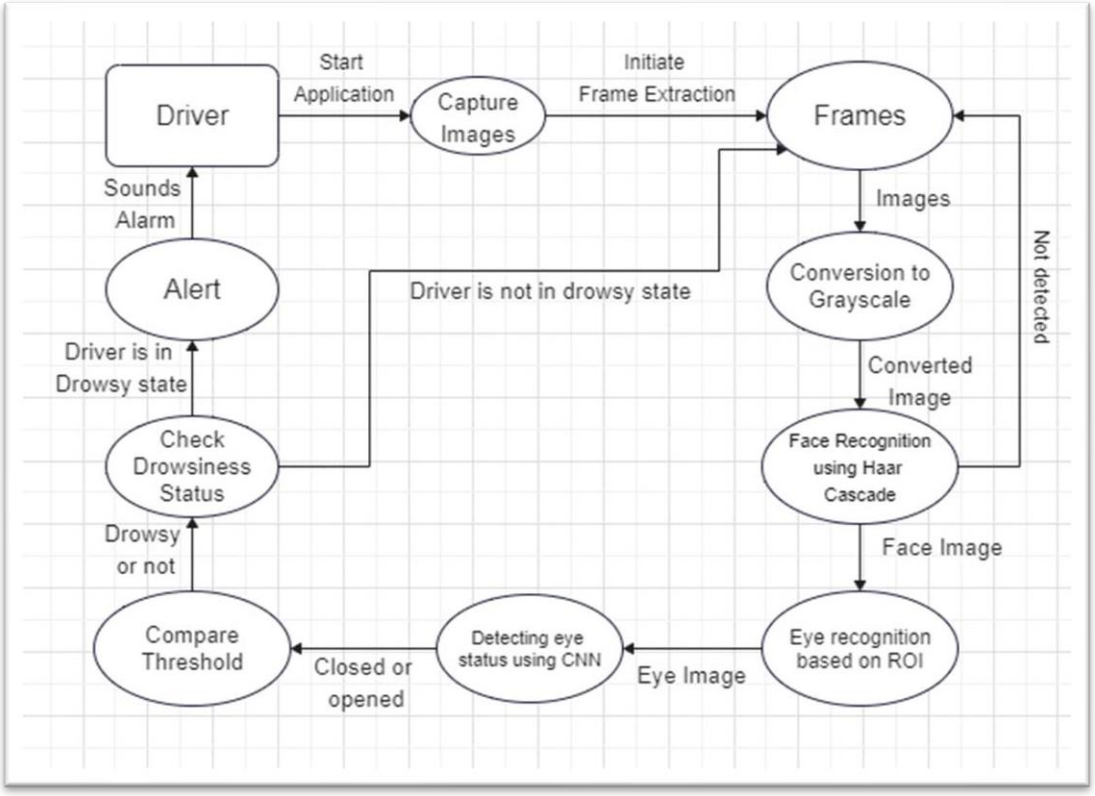


Fig. 7. Level - 2 DFD Representation

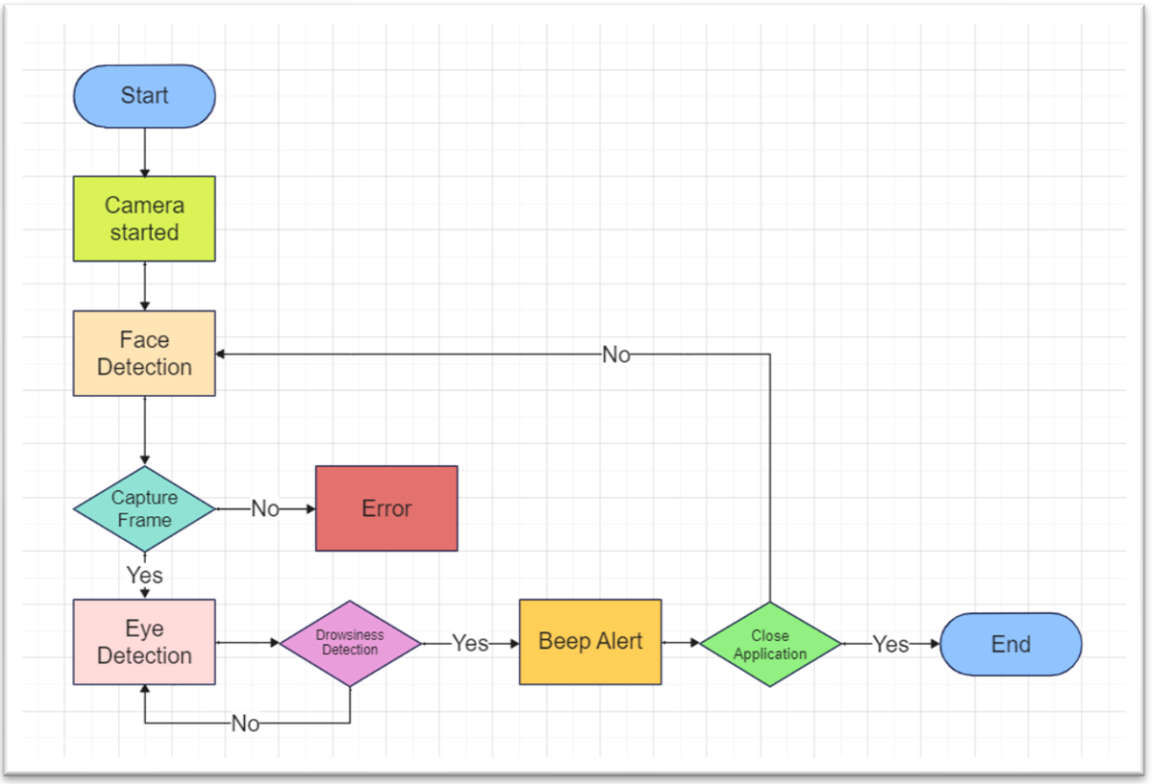


Fig. 8. Flow Chart Representation

4. Results

The successful implementation and testing of this system occurred under specific conditions, demonstrating favorable responsiveness. Notably, the model exhibited effective performance during testing, producing satisfactory outcomes. Below are snapshots illustrating the system's functionality and operation. These images capture key moments during the testing phase, providing visual evidence of the effectiveness of system in action.

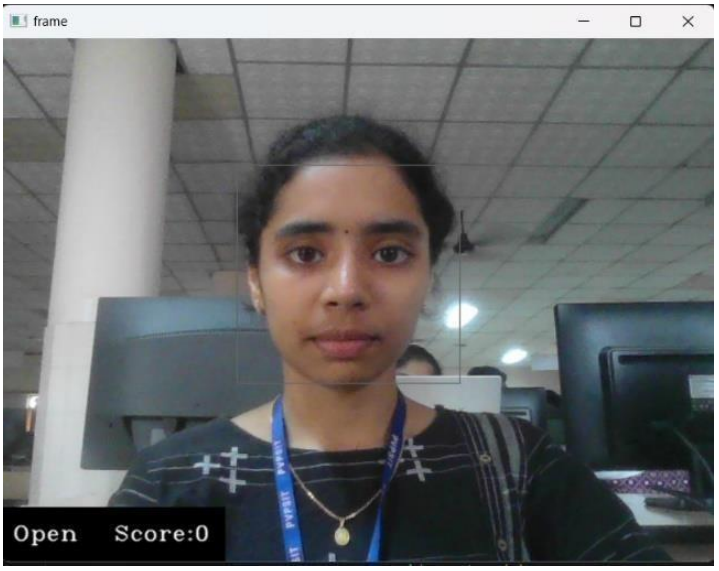


Fig. 9. Eyes in open state

The facial detection process is activated, prompting the system to indicate an initial open score of 0, suggesting that the eyes detected are currently recognized as being in an open state. This functions as the starting point for monitoring the driver's eye movements and assessing drowsiness levels.

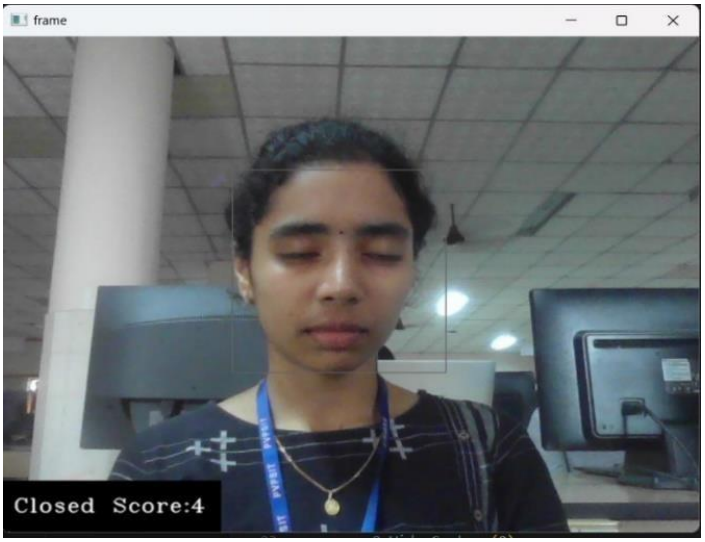


Fig. 10. Eyes closed, not yet at threshold.

In the image below, the detected eyes are marked as closed with a score of 4, indicating their current state of closure. However, the alarm remains inactive as the score falls below the predetermined threshold. While the eyes closure suggests a potential drowsy state, it has not exceeded the threshold required to trigger an alert according to the system's settings.

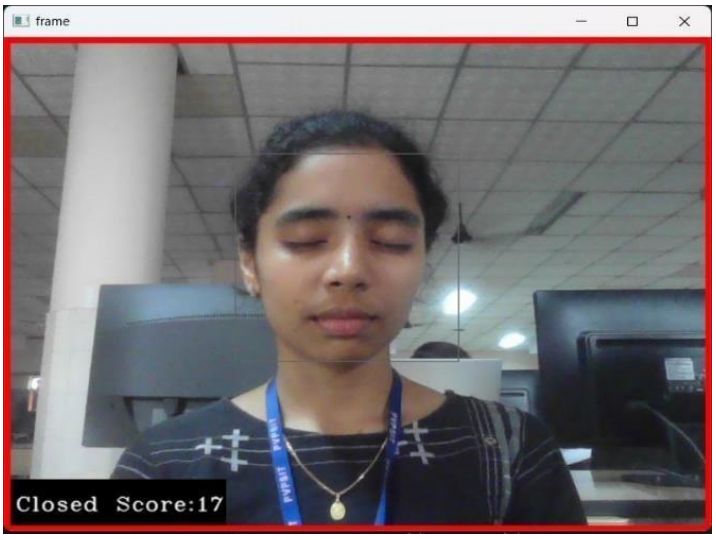


Fig. 11. Eyes closed beyond set threshold.

In the image above, the eyes detected are identified as closed, with a corresponding closed score of 17 indicating prolonged closure. Consequently, the alarm is activated as the score surpasses the predetermined threshold, signaling a potential state of drowsy or fatigue that requires immediate attention. This alarm sounds until the user opens their eyes. When the user opens eyes for a particular period alarm sound will be turned off automatically and the process repeats.

The results produced by this driver drowsiness system are very accurate and can be useful for preventing road accidents. In this, Driver will be continuously monitored to reduce the risks of accidents.

5. Conclusion

In conclusion, the implementation of a driver drowsiness alert system integrating Haar cascade classifiers, OpenCV, Pygame, Keras, and Convolutional Neural Networks (CNN) offers a promising solution to reduce the harm posed by sleepy driving. By leveraging computer vision methods, the system efficiently detects facial features and monitors eye movements in real-time, crucial for recognizing indicators of a driver fatigue. The usage of CNNs enhances the efficiency of drowsiness classification, enabling timely alerts when the eyes of driver exhibit prolonged closure, indicative of sleepiness.

With the incorporation of Pygame, the system delivers audible alerts in the form of alarms, effectively notifying the when a driver exhibits signs of sleepiness. This proactive approach serves to prompt immediate corrective action, potentially preventing harmful accidents caused by drowsy driving. Furthermore, the modular design and incorporation of widely-used libraries like OpenCV and Keras ensure flexibility and scalability, facilitating easy deployment across different platforms and environments.

In essence, the driver drowsiness alert system represents a critical advancement in automotive safety technology, offering a reliable means to enhance driver consciousness and lessen the chance of sleepy driving - related incidents. Through continued research and refinement, such systems have the potential to significantly contribute to safety and save lives by addressing the dangers posed by driver sleepiness.

6. Future Scope

While this project currently functions as a basic model for detecting drowsiness based on eye movements, A scope exists for future enhancements. These characteristics could be integrated into the model to improve its functionality:

- **Real-time tracking**: Current system limitations only allow for analyzing camera feed images. A more advanced approach could incorporate real-time tracking to monitor driver behavior, such as looking around or engaging in other tasks, providing deeper insights into their state of alertness.
- **Integration with other sensors**: To improve the accuracy of drowsiness prediction, integrating the system with additional sensors like heartbeat monitors and steering sensors could offer a more thorough comprehension of the driver's circumstances.
- **Enhanced precision**: Implementing advanced techniques beyond simple classifiers for recognizing the difference between closed and open eyes could significantly improve result accuracy, leading to more reliable drowsiness detection.
- **Personalization**: Customizing the system depending on individual elements such as age, gender, and medical conditions allows for tailored alerts and interventions, optimizing effectiveness for each driver.
- **Mobile application**: Developing a companion mobile application that connects to the system provides drivers with real-time advice on when to take breaks and insights into their fatigue levels, promoting preventative actions to stop sleepy driving.
- **Integration with vehicles:** Allowing the system to interact directly with vehicle controls, such as automatically slowing down or stopping the car when the driver becomes excessively drowsy, enhances safety measures and prevents potential accidents.

7. References

[1] D. MOHAN, “Road Accidents in India,” IATSS Res., vol. 33, no. 1, pp. 75–79, 2014.

[2] S. Mehta, S. Dadhich, S. Gumber, and A. J. Bhatt, “Real-Time Driver Drowsiness Detection System Using Eye Aspect Ratio and Eye Closure Ratio,” pp. 1333–1339, 2019.

[3] Wanghua Deng, Ruoxue Wu. Real-Time Driver-Drowsiness Detection System Using Facial Features. IEEE Access PP(99):1-1.

[4] Chen, Y.W., Kubo, K.: A robust eye detection and tracking technique using Gabor filters. In: Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing, IEEE, vol. 1, pp. 109–112 (2007).

[5] Francois Chollet, Deep Learning with Python, Second Edition

[6] Sumit Saha, A Comprehensive Guide to Convolutional Neural Networks — the ELI5 Way

[7] V.E. Machaca, J. P. Cahuana Nina, and K.M. Fernandez Fabian. A Survey on Drowsiness Detection Techniques. Vol 2747, Paper14

[8] Rizul Sharma, Pratyush Agarwal. Driver drowsiness detection system - ResearchGate

[9] M. Abadi et al.,TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems

[10] DataFlair, AI with Python - Computer Vision Tutorials

[11] Paul Viola and Michael Jones, Rapid Object Detection using a Boosted Cascade of Simple Features

[12] Arcaya-Jordan, A., Pegatoquet, A., & Castagnetti, A. (2019, March). Smart Connected Glasses for Drowsiness Detection: a System-Level Modeling Approach. In 2019 IEEE Sensors Applications Symposium (SAS) (pp.1-6).

[13] PythonProgramming.net, Adding Sounds and Music with Pygame

[14] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR).

[15] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do & Kaori Togashi. Convolutional neural networks: an overview and application in radiology (2018)

[16] S. Vitale, A. Paola, and F. Sorbello, "Bright Pupil Detection in an Embedded, Real-time Drowsiness Monitoring System", in 24th IEEE International Conference on Advanced Information Networking and Applications, 2010

[17] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms", IEEE Transactions on Systems, Man and Cybernetics, pp. 62-66, 1979.

[18] S. Hu and G. Zheng, “Driver drowsiness detection with eyelid related parameters by Support Vector Machine,” International Journal of Expert Systems with Applications, vol. 36, 2009, pp. 7651–7658, doi: http://dx.doi.org/10.1016/j.eswa.2008.09.030C.

[19] Driver Drowsiness Detection using Eye-Closeness Detection (2016 12th International Conference on Signal-Image Technology & Internet-Based Systems).

[20] Hong, Tianyi, Huabiao Qin, and Qianshu Sun. "An improved real-time eye state identification system in driver drowsiness detection," in Proc. IEEE Int. Conf. on Control and Automation, IEEE, 2007, pp. 1449- 1453.

MAIL COMMUNICATION