DRIVER DROWSINESS DETECTION USING DEEP LEARNING

A PROJECT REPORT

submitted by

AKHIL SUDHAN

KTE18MCA007

to

the APJ Abdul Kalam Technological University

in partial fulfillment of the requirements for the award of the degree

of

Master of Computer Applications

Department of Computer Applications

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

(Government Engineering College)

KOTTAYAM - 686 501, KERALA

July 2021

DECLARATION

I undersigned hereby declare that the project report ”DRIVER DROWSINESS DETECTION USING DEEP LEARNING” , submitted for partial fulfillment of the requirements for the award of degree of Master of Computer Applications of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by me under supervision of Asst Prof.Sreelekshmi KR. This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

KOTTAYAM
July 8, 2021

AKHIL SUDHAN

DEPARTMENT OF COMPUTER APPLICATIONS

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

KOTTAYAM

CERTIFICATE

This is to certify that the report entitled **'DRIVER DROWSINESS DETECTION USING DEEP LEARNING'** submitted by **'Mr.Akhil Sudhan [KTE18MCA007]'** to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree of **Master of Computer Applications** is a bonafide record of the project work carried out by him/her under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Internal Supervisor **External Supervisor**

External Examiner **HEAD OF THE DEPARTMENT**

Tecgenix

Date 10-06-21

This is to certify that Mr Akhil Sudhan (KTE18MCA007) , MCA 6th semester student of Rajiv Gandhi Institute of Technology ,Kottayam has successfully completed Internship on the project " Driver Drowsiness Detection using Deep Learning" in python from April 15th to June 10th at TECGENIX Pvt Ltd During the period of his internship programme with us he was found Sincere, punctual ,hardworking ,efficient and Inquisitive .

We wish a bright future

With Regards

HR Manager

+919048245318

www.tecgenix.com

tecgenix@gmail.com

TL-G102, Nazland building,
Near kairali TV office
Kunnukuzhi lane palayam
Trivandrum 33 kerala

TECGENIX

ACKNOWLEDGEMENT

I wish to thank the multitude of people who have helped me during the course of the MCA.First of all, I thank God almighty for His grace and blessings for without his unforeseen guidance this would have remained in dreams.

I express my sincere gratitude to **Dr. Jalaja M.J ,Principal**, Rajiv Gandhi Institute of Technology, Kottayam, for providing the ambiance for carrying out the work of this project.

I deeply indebted to **Prof.John C John**, Head of the Department, Computer Applications and Engineering,for providing permission and availing all required facilities for undertaking the project in a systematic way.

I feel deeply honoured in expressing my sincere thanks to **Asst Prof.Sreelekshmi KR** Assistant Professor of department of Computer Application for the constructive suggestions and inspirations that helped me during the preparation of this project.

I take this opportunity to thank all the technical staffs of Department of Computer Applications for their help.

I also express my gratitude to **Ms.Subilal** , AI Developer,TECGENIX Thiruvanathapuram for providing me with adequate facilities, Support and guidance ways and means to complete this internship.Gratitude may be extended to my parents and friends who supported us for the project.

AKHIL SUDHAN

ABSTRACT

This project describes a real-time system for monitoring driver vigilance. This project is based on Eye Aspect Ratio (EAR), yawning detection and detect drowsiness by comparing its instantaneous value with a previously configured value.We propose a generalised approach using Convolution Neural Networks (CNN),Haar Cascade Algorithm (HCC) and Hidden Markov Model (HMM) in this project. Our project tracks the driver's eyes and feeds it into a pre-trained that predicts the state of the eye. Once the prediction is obtained, we would be able to detect if the driver is drowsy or not. The main components of our system include a camera, for real time image acquisition, a processor for running algorithms to process the acquired image and an alarm to warn the driver when the symptoms are detected in order to avoid potential accidents.The Project Going to be Implemented in Python Language.

TABLE OF CONTENTS

ACKNOWLEDGEMENT i

ABSTRACT ii

LIST OF FIGURES v

CHAPTER 1. Introduction 1

1.1 Need for the Project 1

1.2 Outline of the Report 1

1.3 Motivation 2

1.4 Scope of the Project 2

CHAPTER 2. REQUIREMENT ANALYSIS AND SPECIFICATION 4

2.1 System Study 4

2.1.1 Existing System 4

2.1.2 Proposed System 4

2.2 System specification 5

2.2.0.1 Hardware Specification 5

2.2.0.2 Software Specification 5

2.3 Software Tools 6

2.3.1 Python 6

2.3.2 Matplotlib 6

2.3.3 SciPy 7

2.3.4 NumPy 7

2.3.5 Open CV 7

2.3.6 Keras 7

2.3.7 Tensorflow 8

2.3.8 Imutils 8

CHAPTER 3. SYSTEM MODELLING **9**
3.1 Introduction 9
3.2 Module Description 9
3.3 Data Flow Diagram 10
3.3.1 DFD Symbols 11
3.4 Block Diagram. 15
3.4.1 Training Block Diagram. 15
3.4.2 Overoll 16

CHAPTER 4. SYSTEM DESIGN **17**
4.1 Introduction 17
4.1.1 Architecture Used For Model Building 17
4.2 Results 18
4.2.1 Data Sets 18
4.2.2 Data Sets 19
4.2.3 Data Sets 19
4.2.4 Training 20
4.2.5 Product Backlog 20
4.2.6 Sprint Backlog 21

CHAPTER 5. SYSTEM TESTING **23**
5.1 Introduction 23
5.1.1 Unit Testing 23
5.1.2 Integration Testing 24
5.1.3 User Acceptance Testing 24
5.2 Testcases 25
5.3 Results 29
5.3.1 Eyes Closed. 29
5.3.2 Drowsy 30
5.3.3 Yawning. 31

CHAPTER 6. SYSTEM IMPLEMENTATION **32**
6.1 Implementation Methods 32
6.2 Implementation Plan 33

CHAPTER 7. CONCLUSION AND FUTURE SCOPE **35**

REFERENCES **36**

LIST OF FIGURES

3.1 Level 0 12
3.2 Level 1.1 12
3.3 Level 1.2 13
3.4 Level 1.2.1 13
3.5 Level 1.3 14
3.6 Training Block Diagram. 15
3.7 Observations 16

4.1 Architecture of VGG16 18
4.2 Normal,Drowsy,Yawning 18
4.3 Normal,Drowsy,Yawning 19
4.4 Model Accuracy 19
4.5 Summary of The Model 20
4.6 Product Backlog 20
4.7 Sprint Backlog 21
4.8 Sprint Backlog 22

5.1 Features 25
5.2 Distance. 26
5.3 With Cooling Glass 26
5.4 Drowsines 27
5.5 Drowsy 27
5.6 Facial Contours 28
5.7 Yawning 28
5.8 Wearning Transperent Glass 28
5.9 Eyes Closed 29
5.10 Drowsy Result 30
5.11 Yawn 31

CHAPTER 1

INTRODUCTION

1.1 Need for the Project

Fatigue and microsleep are the reasons behind many severe road accidents. These can be avoided if the symptoms of fatigue are detected on time This project describes a real-time system for monitoring driver vigilance.. Our project tracks the driver's eyes and feeds it into a pre-trained that predicts the state of the eye. Once the prediction is obtained, we would be able to detect if the driver is drowsy or not. The main components of our system include a camera, for real time image acquisition, a processor for running algorithms to process the acquired image and an alarm to warn the driver when the symptoms are detected in order to avoid potential accidents.

1.2 Outline of the Report

Details of proposed system is provided in chapter 2. Hardware and software specifications for both development and implementations are detailed. Module description and data flow diagrams are described in chapter 3. Chapter 4 includes the database design and form design. Also, the screenshots of form are given in chapter 4. Various types of tests and implementation details are detailed. The future scope and conclusions are summarized in chapter 5.

1.3 Motivation

Our current statistics reveal that just in 2018 in India alone, 148,707 people died due to car related accidents. Of these, at least 21 percent were caused due to fatigue causing drivers to make mistakes. This can be a relatively smaller number still, as among the multiple causes that can lead to an accident, the involvement of fatigue as a cause is generally grossly underestimated. Fatigue combined with bad infrastructure in developing countries like India is a recipe for disaster. Fatigue, in general, is very difficult to measure or observe unlike alcohol and drugs, which have clear key indicators and tests that are available easily. Probably, the best solutions to this problem are awareness about fatigue-related accidents and promoting drivers to admit fatigue when needed. The former is hard and much more expensive to achieve, and the latter is not possible without the former as driving for long hours is very lucrative. When there is an increased need for a job, the wages associated with it increases leading to more and more people adopting it. Such is the case for driving transport vehicles at night. Money motivates drivers to make unwise decisions like driving all night even with fatigue. This is mainly because the drivers are not themselves aware of the huge risk associated with driving when fatigued. Some countries have imposed restrictions on the number of hours a driver can drive at a stretch, but it is still not enough to solve this problem as its implementation is very difficult and costly.

1.4 Scope of the Project

In 2018, there were around 151 thousand deaths due to road accidents in India. Of these, at least 21 percent were caused due to fatigue causing drivers to make mistakes. This can be a relatively smaller number still, as among the multiple causes that can lead to an accident, the involvement of fatigue as a cause is generally grossly

underestimated. Fatigue combined with bad infrastructure in developing countries like India is a recipe for disaster. Fatigue, in general, is very difficult to measure or observe unlike alcohol and drugs, which have clear key indicators and tests that are available easily. Probably, the best solutions to this problem are awareness about fatigue-related accidents and promoting drivers to admit fatigue when needed. The former is hard and much more expensive to achieve, and the latter is not possible without the former as driving for long hours is very lucrative. When there is an increased need for a job, the wages associated with it increases leading to more and more people adopting it. Such is the case for driving transport vehicles at night. Money motivates drivers to make unwise decisions like driving all night even with fatigue. This is mainly because the drivers are not themselves aware of the huge risk associated with driving when fatigued. Some countries have imposed restrictions on the number of hours a driver can drive at a stretch, but it is still not enough to solve this problem as its implementation is very difficult and costly.

CHAPTER 2

REQUIREMENT ANALYSIS AND SPECIFICATION

2.1 System Study

System analysis was done by having frequent meetings with our clients and thereby acquiring a detailed specification of their Requirement.

We had several meetings with one of our client.Their main requirement was to Make a System inoder to save drivers and co passengers lives.We Were prepared some questionnaires regarding our doubts and cleared it with them.

2.1.1 Existing System

The present system is all about manually Waking up the driver or by using the safety measures provided by the car manufracturer.No Alert System softwares Present till now.

2.1.2 Proposed System

Driver Drowsiness Detection is a Software. This Software based on Eye Aspect Ratio (EAR), yawning detection and detect drowsiness by comparing its instantaneous value with a previously configured value We propose a generalised approach using Convolution Neural Networks (CNN),Cascade Algorithm (HCC) and Hidden Markov Model (HMM) in this project Our project tracks the driver's eyes and feeds it into a pre-trained that predicts the state of the eye. The main components of our system

include a camera, for real time image acquisition, a processor for running algorithms.

2.2 System specification

2.2.0.1 Hardware Specification

- Processor : i3,1.8 GHz
- RAM : 4GB MB (or greater)
- Harddisk Drive :30GB
- Video :800*600,256 Colors
- CD Rom: Required
- Processor: Intel Core i3
- Family Monitor :Proper Resolution
- Display Adapter :SMC Ethernet card Elite
- Camera:180p Resolution

2.2.0.2 Software Specification

- Operating System :Windows7 and Above
- Front End;Python
- Platform : Anaconda(Spyder)/Colab

2.3 Software Tools

2.3.1 Python

Python 3.8 used in this project. Python is an interpreted, high-level, generalpurpose programming language. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde Informatica (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL), capable of exception handling and interfacing with the Amoeba operating system.Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspectoriented programming (including by meta programming and meta objects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming. Python 2.7 is the last major release in the 2.x series, as the Python maintainers have shifted the focus of their new feature development efforts to the Python 3.x series[4]. This means that while Python 2 continues to receive bug fixes, and to be updated to build correctly on new hardware and versions of supported operated systems, there will be no new full feature releases for the language or standard library.

2.3.2 Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib.pyplot is a collection of command style functions that make matplotlib work like MATLAB

2.3.3 SciPy

SciPy is a free and open-source Python library used for scientific computing and technical computing. It uses NumPy for more mathematical functions. SciPy uses NumPy arrays as the basic data structure, and comes with modules for various commonly used tasks in scientific programming, including linear algebra, integration (calculus), ordinary differential equation solving, and signal processing.

2.3.4 NumPy

NumPy is a python library used for working with arrays. It also has functions for working in the domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely. NumPy stands for Numerical Python.

2.3.5 Open CV

OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

2.3.6 Keras

Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load : it offers consistent simple APIs, it minimizes the number of user actions required for common use cases , and it provides clear actionable error messages. It also has extensive documentation and developer guides. Built on top of TensorFlow 2.0, Keras is an industry-strength framework that can scale

to large clusters of GPUs or an entire TPU pod . It's not only possible ; it is easy.

2.3.7 Tensorflow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library , and is also used for machine learning applications such as neural networks. TensorFlow provides stable Python (for version 3.7 across all platforms) and C APIs and without API backwards compatibility guarantee : C++, Go, Java, JavaScript and Swift. Third-party packages are available for C, Haskell, Julia, MATLAB, R, Scala, Rust OCaml, and Crystal

2.3.8 Imutils

Imutils A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, displaying Matplotlib images, sorting contours, detecting edges, and much more easier with OpenCV and both Python 2.7 and Python 3. This package includes a series of OpenCV + convenience functions that perform basics tasks such as translation, rotation, resizing, and skeletonization.

CHAPTER 3

SYSTEM MODELLING

3.1 Introduction

System modeling is the inter disciplinary study of the model to conceptualize and construct in business and IT development.

A common type of systems modeling is function modeling,with specific techniques such as the Data Flow Diagram.

These models can be extended using functional decomposition,and can be linked to requirements models for further systems partition.

3.2 Module Description

- MODULES:

- Image Capturing Module

  This is the stage where image frames are taken from a fixed camera. The image frames are taken in such a manner that only the face of the driver is captured

- Face Detection:-

  The second stage typically aims to detect the face in the image frames. From the image frames, the face is detected first. Convolutional Neural Network (CNN) feeds the whole image to a network that has multiple filters and the face features

are extracted from this network.

- Feature Extraction Module:-

  If face detection is applied, features are usually extracted using different methods such as landmark localization, Histogram of oriented gradients (HOG), and Local Binary Patterns (LBP). This step simplifies the image by extracting useful information and discarding irrelevant information. Eyes detected by pixel difference, or by using Sobel vertical edge operator.

3.3 Data Flow Diagram

A data flow diagram is graphical tool used to describe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations.

A full description of a system actually consists of a set of data flow diagrams. Using two familiar notations Yourdon, Gane and Sarson notation develops the data flow diagrams. Each component in a DFD is labeled with a descriptive name. Process is further identified with a number that will be used for identification purpose. The development of DFD'S are done in several levels. Each process in lower level diagrams can be broken down into a more detailed DFD in the next level.

The lop-level diagram is often called context diagram. It consists a single

process bit, which plays vital role in studying the current system. The process in the context level diagram is exploded into other process at the first level DFD. The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done until further explosion is necessary and an adequate amount of detail is described for analyst to understand the process. Larry Constantine first developed the DFD as a way of expressing system requirements in a graphical from, this lead to the modular design.

3.3.1 DFD Symbols

In DFD, there are four symbols:

1. A square defines a source (originator) or destination of system data.
2. An arrow identifies data flow. It is the pipeline through which the information flows.
3. A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.
4. An open rectangle is a data store, data at rest or a temporary repository of data

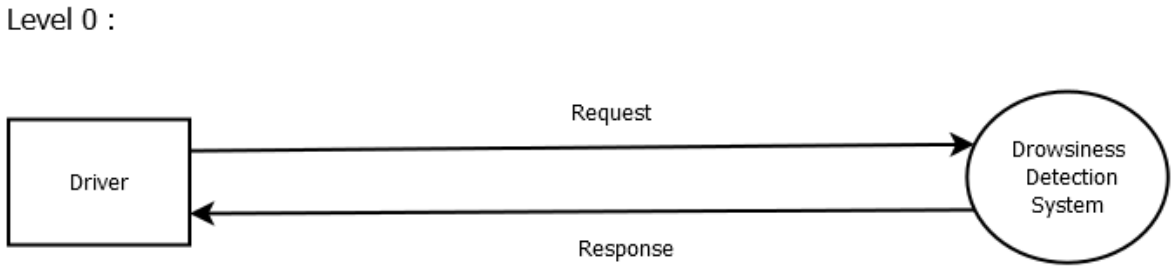

Fig. 3.1. Level 0

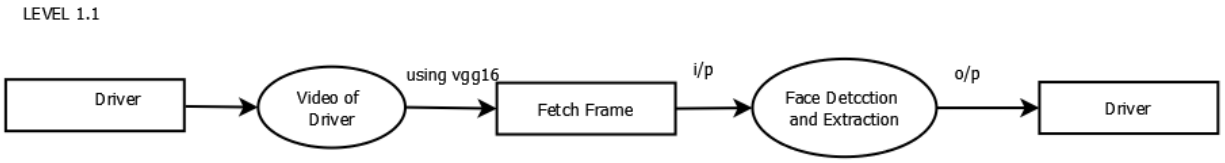

Fig. 3.2. Level 1.1

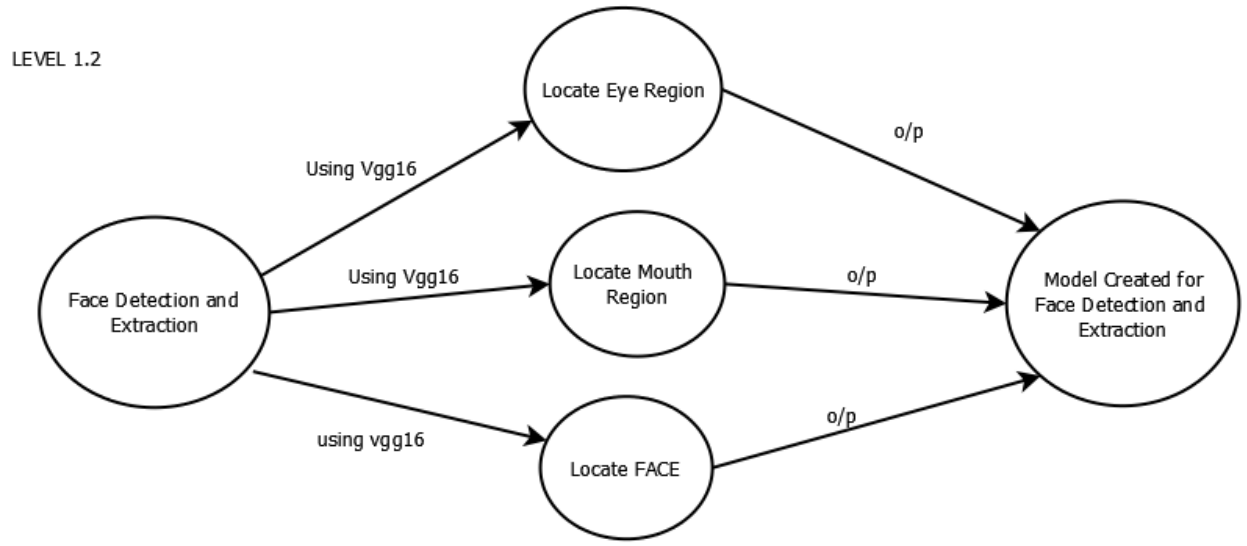

Fig. 3.3. Level 1.2

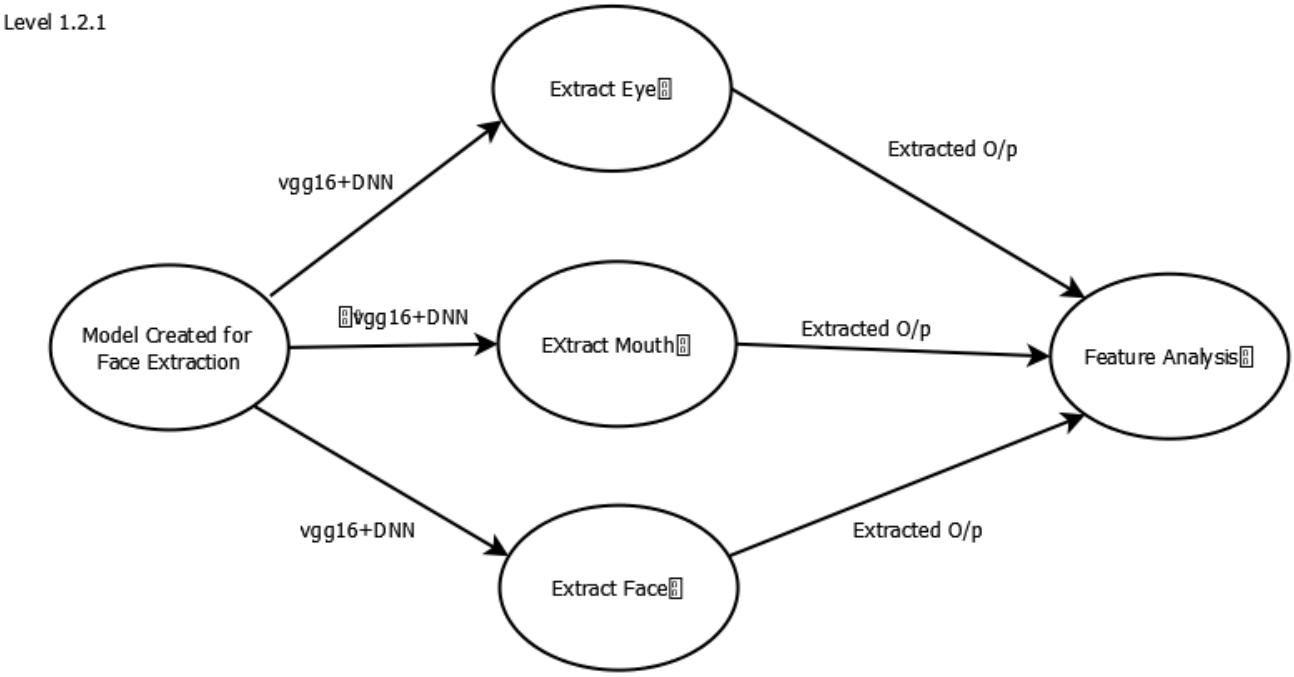

Fig. 3.4. Level 1.2.1

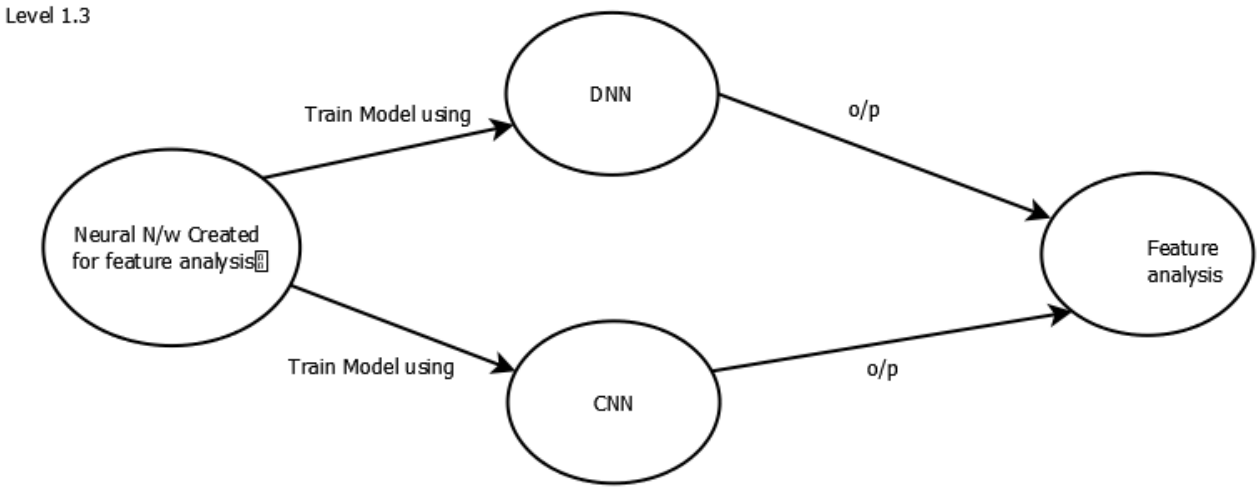

Fig. 3.5. Level 1.3

3.4 Block Diagram.

3.4.1 Training Block Diagram.

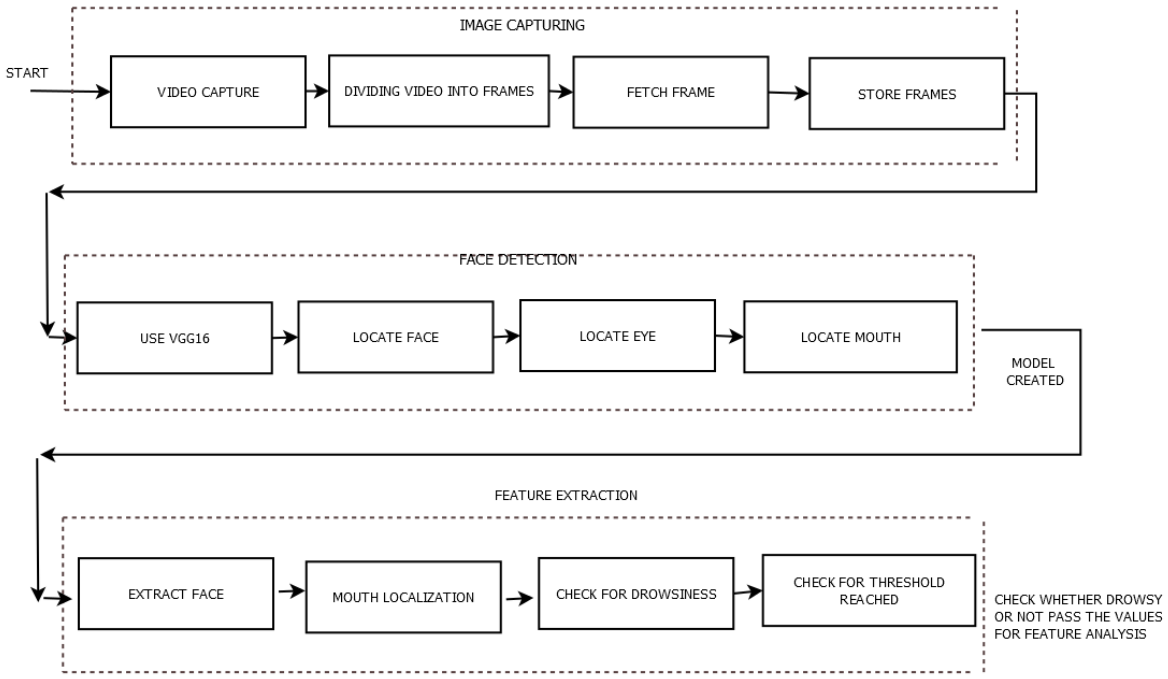

Fig. 3.6. Training Block Diagram.

3.4.2 Overoll

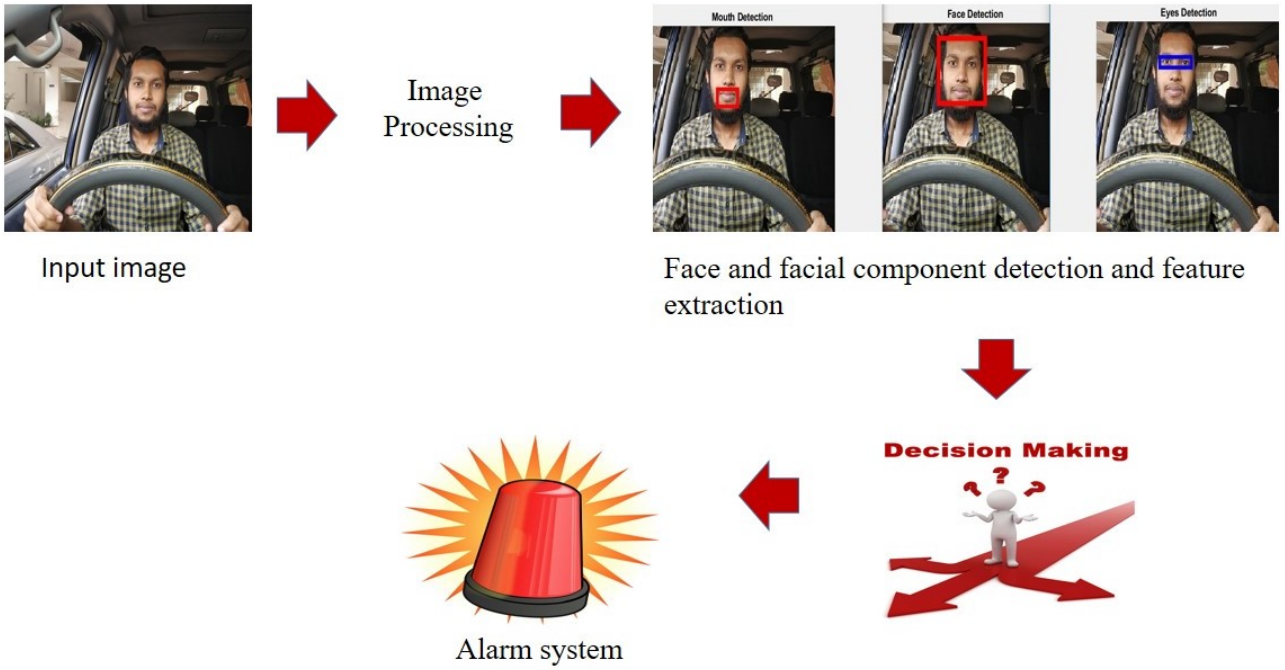

Fig. 3.7. Observations

CHAPTER 4

SYSTEM DESIGN

4.1 Introduction

Machine learning continues to be an increasingly integral component of our lives, whether we're applying the techniques to research or business problems. Machine learning models ought to be able to give accurate predictions in order to create real value for a given organization.

The amount that the weights are updated during training is referred to as the step size or the "learning rate". Specifically, the learning rate is a configurable hyper parameter used in the training of neural networks that has a small positive value, often in the range between 0.0 and 1.0. We trained the model with different learning rate and thus obtained the following results

4.1.1 Architecture Used For Model Building

VGG16 is a convolution neural net (CNN) architecture which was used to win ILSVR(Imagenet) competition in 2014. It is considered to be one of the excellent vision model architecture till date. Most unique thing about VGG16 is that instead of having a large number of hyper-parameter they focused on having convolution layers of 3x3 filter with a stride 1 and always used same padding and maxpool layer of 2x2 filter of stride 2. It follows this arrangement of convolution and max pool layers consistently throughout

the whole architecture. In the end it has 2 FC(fully connected layers) followed by a softmax for output. The 16 in VGG16 refers to it has 16 layers that have weights. This network is a pretty large network and it has about 138 million (approx) parameters

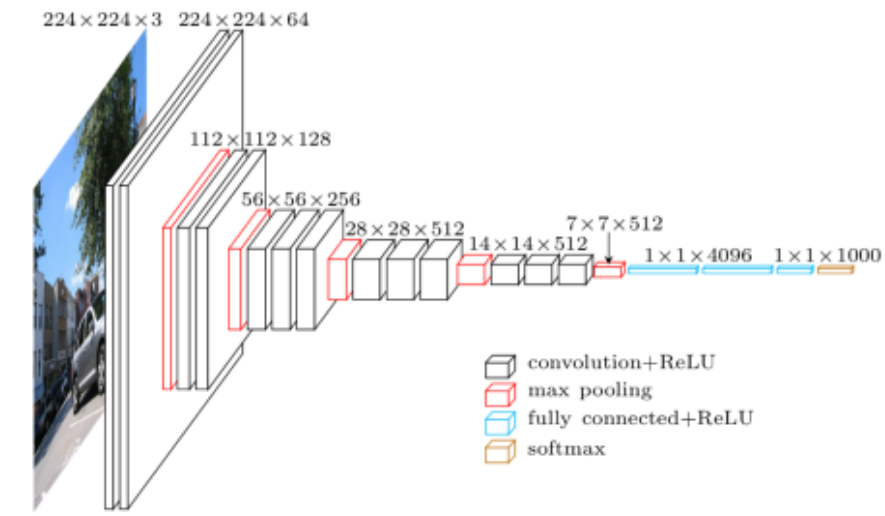

Fig. 4.1. Architecture of VGG16

4.2 Results

4.2.1 Data Sets

Fig. 4.2. Normal,Drowsy,Yawning

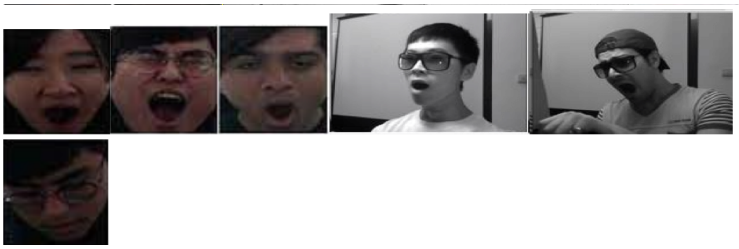

Fig. 4.3. Normal,Drowsy,Yawning

4.2.2 Data Sets

The graph itself showing some abnormalities in loss value and accuracy hence it is not an ideal one.

4.2.3 Data Sets

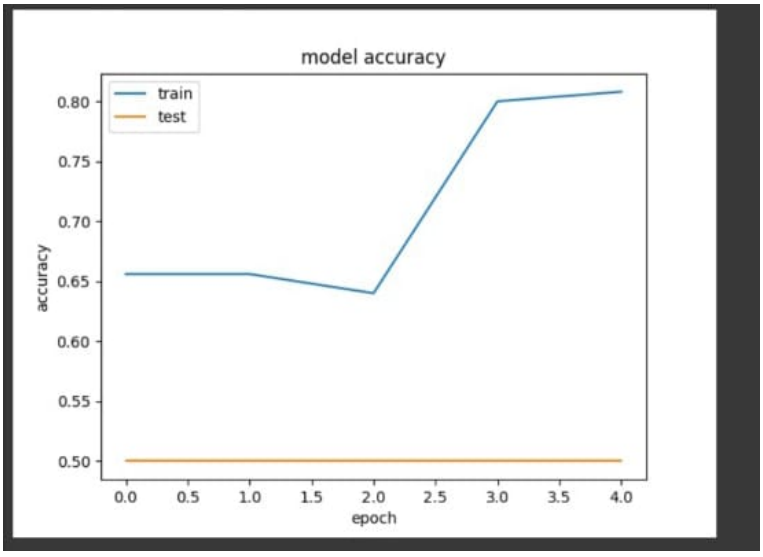

Fig. 4.4. Model Accuracy

Model Accuracy

4.2.4 Training

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808

Fig. 4.5. Summary of The Model

4.2.5 Product Backlog

SL NO	DESCRIPTION	PRIORITY
1	Data Set Collection	1
2	Installation of Packages	2
3	Model Building	3
4	Model Training and Learning	4
5	Cross Validation And Back Propagation	5
6	Model Testing and Matching	6

Fig. 4.6. Product Backlog

4.2.6 Sprint Backlog

SPRINT	DATE	SPRINT GOAL	STATUS
1	12-04-21	Find Suitable Project Topic	Completed
2	13-04-21	Topic discussion	Completed and in Progress
3	14-04-21	Discussion About Modules	Completed
4	14-04-21-	Prepare Product Backlog	Partially Completed
5	14-04-21-	Prepare Sprint Backlog	Partially Completed
6	15-04-21 TO 22-04-21	Discussion About Datasets	Partially Completed
7	23-04-21 TO 26-04-21	Detailed Study About Project	Completed
8	27-04-21 TO 30-04-21	Dataset Collection	Completed
9	31-04-21 TO 5-05-21	Literature Study	Partially Completed
10	6-05-21 TO 12-05-21	Architecture Study	Completed

Fig. 4.7. Sprint Backlog

11	13-05-21 TO 18-05-21	Model Creation	Completed
12	19-05-21	Selecting Algorithm suitable for working model	Partially Completed
13	20-05-21 TO 28-05-21	Model Building	Completed
14	29-05-21 TO 8-06-21	Model training	Completed
15	9-06-21 TO 12-06-21	Model Learning	Completed
16	13-06-21 TO 19-06-21	Model testing and matching	Completed and in progress

Fig. 4.8. Sprint Backlog

CHAPTER 5

SYSTEM TESTING

5.1 Introduction

System testing is an expensive but critical process that can take as much as 50 percent of budget for program development, the common view of testing hold by users that is performed to prove that there is no error in the program. However, this virtually impossible since analysis cannot prove that software is free and clear to errors. Testing is the process of excecuting a program with explicit intension of finding errors. If testing is conducted successfully, it will uncover in the software as the secondary benefits.

5.1.1 Unit Testing

Unit Testing of software applications is done during the development (coding) of an application. The objective of Unit Testing is to isolate a section of code and verify its correctness. In procedural programming, a unit may be an individual function or procedure. Under the unit testing,

- Writes a section of code in the application just to test the function.
- Create Test Cases.
- Review / Rework.
- Baseline.

- Execute Test Cases.

5.1.2 Integration Testing

Data can be lost across any interface, one module can have an adverse effect on another, sub functions when combined, may not produce the desired major functions. Integration testing is a systematic testing to discover errors associated within the interface. The objective is to take unit tested modules and build a program structure. All the modules are combined and tested as a whole. Here the Server module and Client module options are integrated and tested. This testing provides the assurance that the application is well integrated functional unit with smooth transition of data. The developer of each program unit identifies and documents the unit's interfaces for the following unit operations:

- External inquiry (responding to queries from terminals for information).
- External input (managing transaction data entered for processing).
- External filing (obtaining, updating, or creating transactions on computer files).
- Internal filing (passing or receiving information from other logical processing.
- External display (sending messages to terminals) units).
- External output (providing the results of processing to some output device or unit).

5.1.3 User Acceptance Testing

UAT is a type of testing performed by the Client to certify the system with respect to the requirements that was agreed upon. User acceptance of a system is the

key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the system users at time of developing and making changes whenever required.

5.2 Testcases

Some of the test cases are given below :

- Detecting Features of Normal Eye.

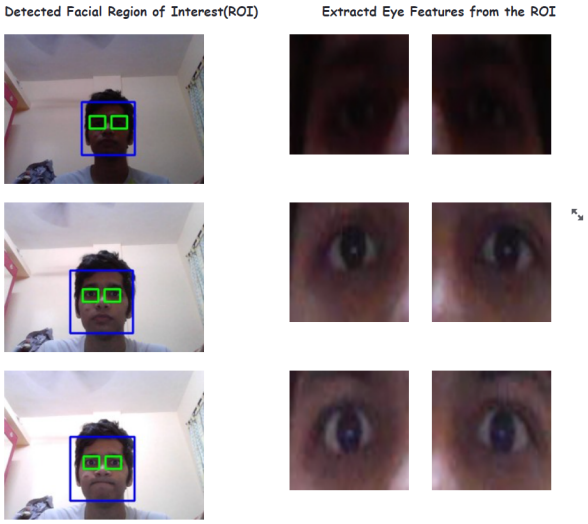

Fig. 5.1. Features

- Eye Distances In Detail

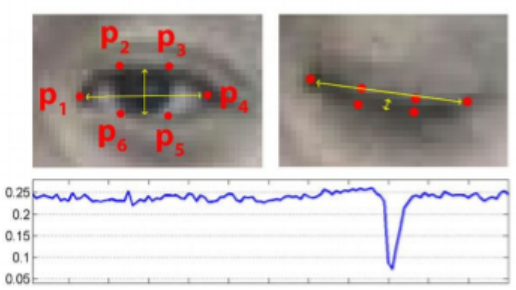

Fig. 5.2. Distance.

- When a Person is Wearing a glass While Drivig.

Fig. 5.3. With Cooling Glass

- When Drowsy.

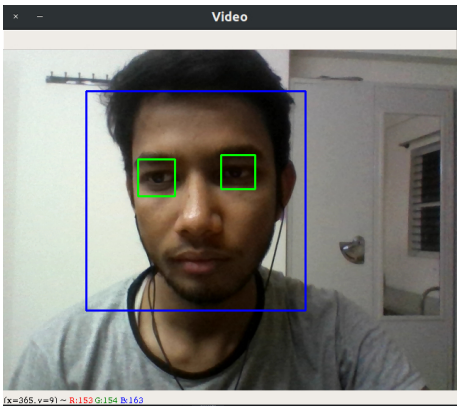

Fig. 5.4. Drowsines

- Drowsy and Eyes Close

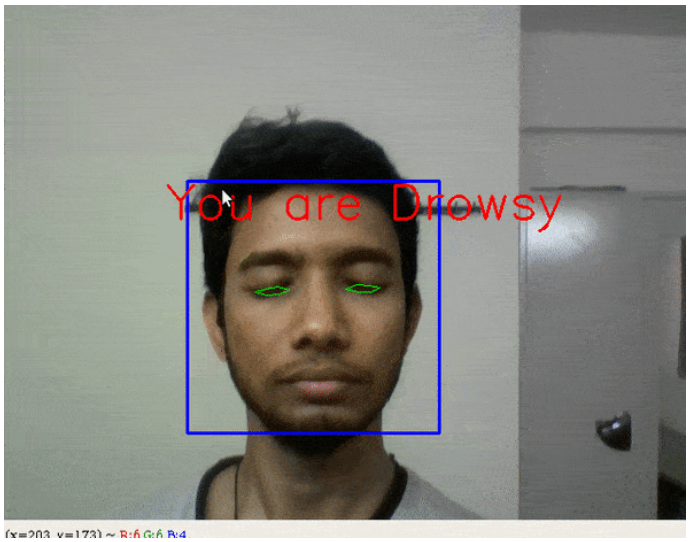

Fig. 5.5. Drowsy

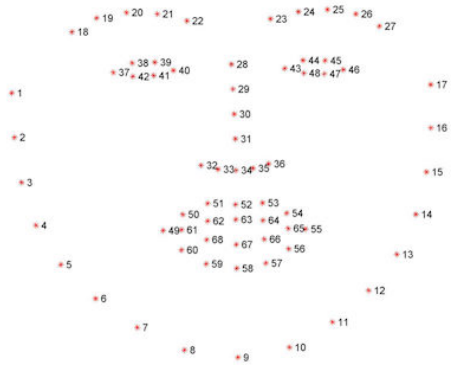

Fig. 5.6. Facial Contours

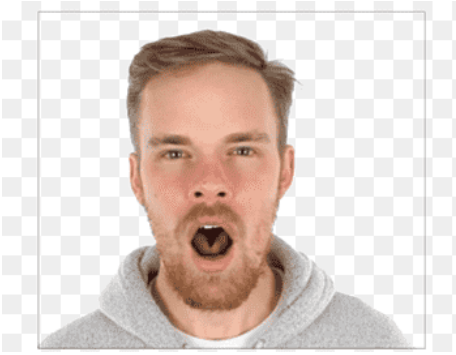

Fig. 5.7. Yawning

Fig. 5.8. Wearning Transperent Glass

5.3 Results

5.3.1 Eyes Closed.

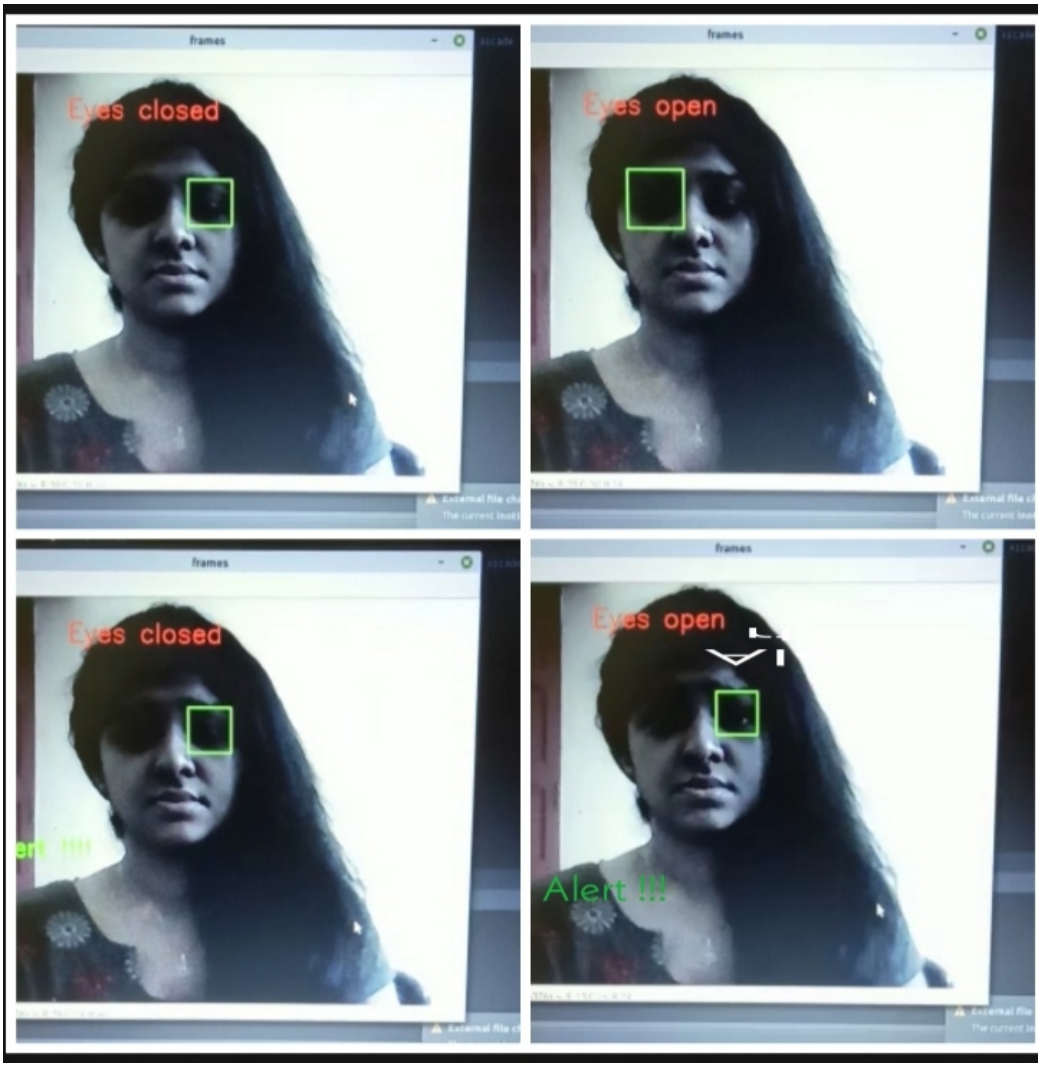

Fig. 5.9. Eyes Closed

5.3.2 Drowsy

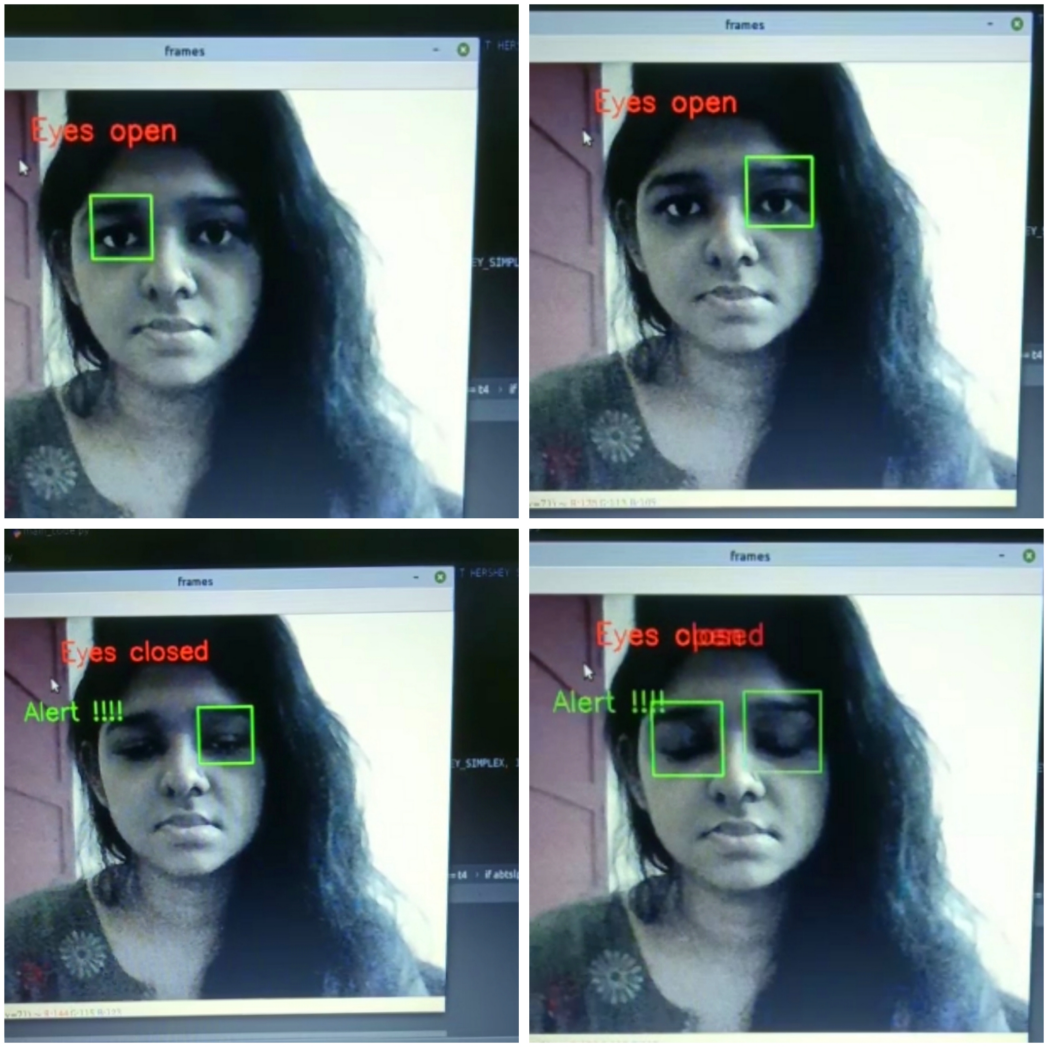

Fig. 5.10. Drowsy Result

5.3.3 Yawning.

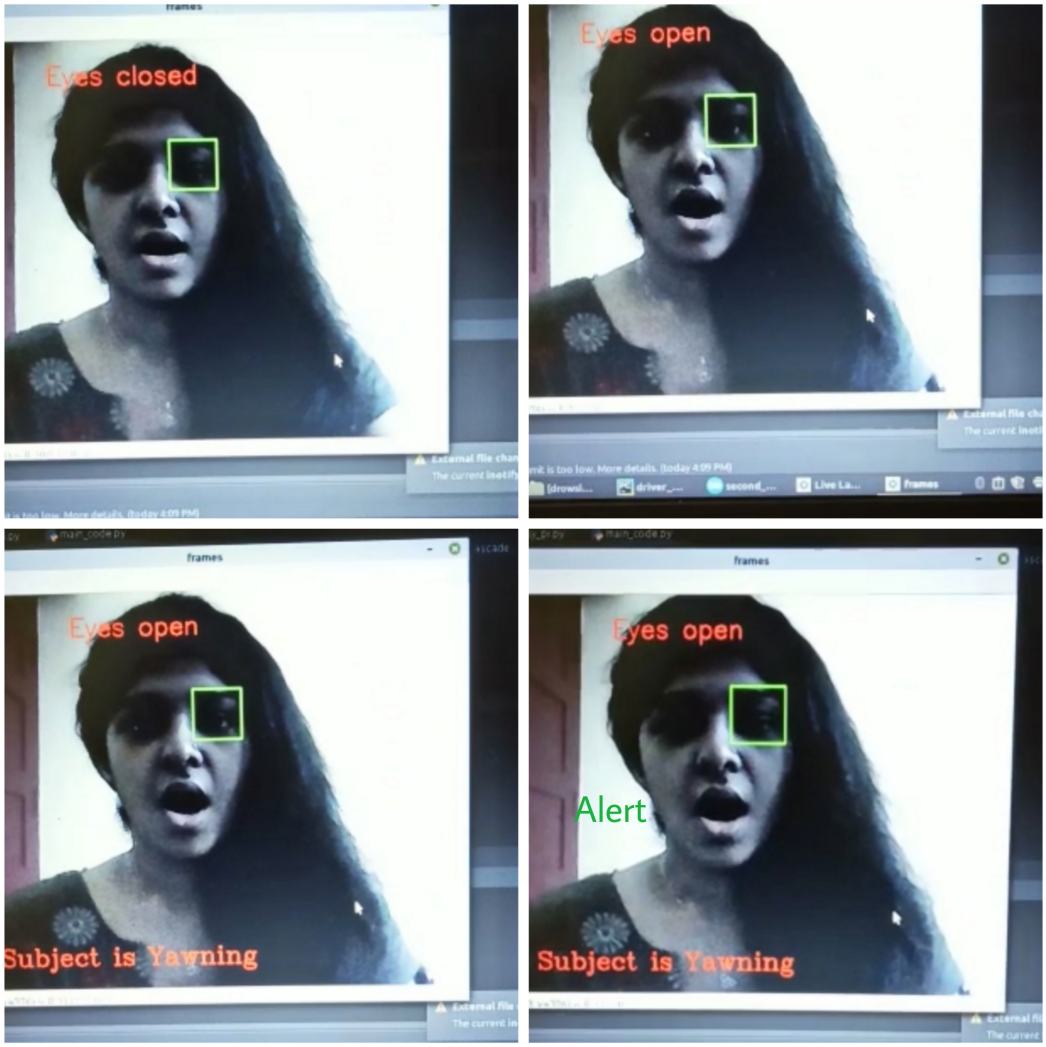

Fig. 5.11. Yawn

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 Implementation Methods

Software implementation refers to the process of adopting and integrating a software application into a business workflow. Implementation of new tools and software into an enterprise can be complex, depending on the size of the organization and the software. Prior to implementation, the software should be selected by assessing needs, budget, potential benefits, obstacles, and so forth. Once the solution is chosen, implementation can begin. For software implementation initiatives to succeed, it's important to follow a few key steps:

- Pilot Program – Prior to full-scale installation, pre-test the software with a test group within the company
- Installation – The IT department should work with the vendor to install the application across all target machines
- Onboarding and Training – Once installed, develop an onboarding program and a software training program to ensure employees are able to utilize all functions and features
- Monitoring, Maintenance, and Follow-Up – Throughout the implementation process, monitor user feedback, usage data, and measure against KPIs

Software implementation can be costly and time-consuming. For this reason, many businesses use digital adoption platforms (DAPs) training and gain insights from app usage data. DAPs provide in-app training and guidance, which significantly reduces human labor time, training time, and employee frustration.Regardless of the software being implemented, businesses should seriously invest time and energy into effective training programs.

Doing so will increase the chances of a successful implementation, boost productivity, and increase employee satisfaction

6.2 Implementation Plan

Implementing new software is always a challenge, even for the most flexible companies. Succeeding demands planning transitions smoothly and providing your team with relevant training material on the new software. implementation process steps:

- Planning Ahead: Always start with a plan! Identify which processes and teams are going to be affected by the implementation and create a shared timeline for carrying the execution gradually. At this stage, you should try and question every possible thing that may get affected by the new software:

- Process Design: Process design is a technique that allows you to organize and run things more efficiently, no matter whether it's a business, software or a team. Most of the commercially available software nowadays has been created using process design methods. This is why when implementing new software you will probably need to adjust many of your new processes to the logic that was used for building that software.

- Solution Design: Once you have created the process design, it is time to work on

the solution design, which is essentially a roadmap of business requirements and processes.

- Configuration and Customization: It is now time to install the software and proceed to configure those features that can be used immediately. As a matter of fact, this step should always come first, before defining any processes or rushing to customize any module.

- Integration: Integration is a critical step within software implementation and it involves migrating data from one system to another. With proper integrations, you can save your team from having to copy data between systems manually. They will thank you for having thought about them

- Reporting: This phase is about understanding what information is valuable for your teams in order to improve their decision-making process on a daily basis.

- Training Testing Training may come in different forms and to different groups, from educating your project team on the new software to teaching the end-user how it works. Like training, testing is also an ongoing process that should be done throughout the software implementation process, but also after the team has been using the software for a while.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

The Driver drowsiness detection and system developed is capable of detecting drowsiness in a rapid manner. The system which can differentiate normal eye blink,Yawning and drowsiness which can prevent the driver from entering the state of sleepiness while driving. The system works well even in case of drivers wearing spectacles and under low light conditions also. During the monitoring, the system is able to decide if the eyes are opened or closed. When the eyes have been closed for about two seconds, the alarm beeps to alert the driver. By doing this many accidents will reduced and provides safe life to the driver and vehicle safety. A system for driver safety and car security is presented only in the luxurious costly cars. Using drowsiness detection system, driver safety can be implemented in normal cars also.

The future works may focus on the utilization of outer factors such as vehicle states, sleeping hours, weather conditions, mechanical data, etc, for fatigue measurement.Monitoring the driver's state of drowsiness and vigilance and providing feedback on their condition so that they can take appropriate action is one crucial step in a series of preventive measures necessary to address this problem. Currently there is not adjustment in zoom or direction of the camera during operation. Future work may be to automatically zoom in on the eyes once they are localized.

REFERENCES

[1] Kai-Wei Ke, Muhammad R. Zulman, Ho-Ting Wu,Yu-Fu Huang. ”Drowsiness Detection System Using Heartbeat Rate in Android-based Handheld Devices”.*2016*.

[2] P Kingston Stanley,Jayaprahash T,Sibin Lal S,P Vijay Daniel.”Embedded based Drowsiness Detection using EEG Signals”*2017*.

[3] Fouzia, Roopalakshmi R, Jayantkumar A Rathod, Ashwitha S Shetty, Supriya k.”Driver Drowsiness Detection System Based on Visual Features”*2018*.

[4] Shigeyuki Tateno,Xia Guan,Rui Cao,Zhaoxian Qu ,”Development of Drowsiness detection system based on respiration changes using Heart rate monitoring”.*2018*.

[5] Marchel T. Tombeng, Hence Kandow, Stenly I. ”Adam Android-Based Application To Detect Drowsiness When Driving Vehicle”.*2020*.