

DIGITAL EYE SCREENING APPLICATION

*Project report submitted to
Visvesvaraya National Institute of Technology,
Nagpur in partial fulfilment of the requirements for
the award of the degree.*

Bachelor of Technology Computer Science and Engineering

by

J.C. Manjunath Reddy

BT20CSE056

Motamarri Sai Roopak

BT20CSE084

Pilli Vishal

BT20CSE096

Sangaraju Naga Sathwik

BT20CSE108

Under the Guidance of

Dr. Ashish Tiwari



Department of Computer Science and Engineering
Visvesvaraya National Institute of Technology
Nagpur 440010 (India)

2024

DIGITAL EYE SCREENING APPLICATION

*Project report submitted to
Visvesvaraya National Institute of Technology,
Nagpur in partial fulfilment of the requirements for
the award of the degree.*

Bachelor of Technology Computer Science and Engineering

by

J.C. Manjunath Reddy

BT20CSE056

Motamarri Sai Roopak

BT20CSE084

Pilli Vishal

BT20CSE096

Sangaraju Naga Sathwik

BT20CSE108

Under the Guidance of

Dr. Ashish Tiwari



Department of Computer Science and Engineering
Visvesvaraya National Institute of Technology
Nagpur 440010 (India)
2024

Department of Computer Science and Engineering
Visvesvaraya National Institute of Technology, Nagpur

DECLARATION

We hereby declare that this project work titled **DIGITAL EYE SCREENING APPLICATION** is carried out by us in the Department of Computer Science and Engineering of Visvesvaraya National Institute of Technology, Nagpur under the guidance of Dr. Ashish Tiwari. The work is original and has not been submitted earlier whole or in part for the award of any degree/diploma at this or any other Institution/University.

Sr. No.	Names	Enrollment No.	Signature
1	J.C. Manjunath Reddy	BT20CSE056	<i>J.C. Manjunath</i>
2	Motamarri Sai Roopak	BT20CSE084	<i>Sai Roopak.</i>
3	Pilli Vishal	BT20CSE096	<i>P.V.</i>
4	Sangaraju Naga Sathwik	BT20CSE108	<i>S. Naga Sathwik</i>

Date: 9/05/2024

**Department of Computer Science and Engineering
Visvesvaraya National Institute of Technology, Nagpur**



CERTIFICATE

This is to certify that the project titled “**Digital Eye Screening Application**”, submitted by **J.C. Manjunath Reddy, Motamarri Sai Roopak, Pilli Vishal and Sangaraju Naga Sathwik** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering**, VNIT Nagpur. The work is comprehensive complete and fit for final evaluation.

Dr. Ashish Tiwari (Project Guide)
Assistant Professor,
Computer Science and Engineering,
VNIT, Nagpur

Dr. M.P Kurhekar
Head of Department,
Department of Computer Science and
Engineering, VNIT, Nagpur

सहायक प्राध्यापक
Assistant Professor
संगणक विज्ञान एवं अभियांत्रिकी विभाग
Dept. of Computer Science & Engg.
वि.रा.प्रौ.सं. नागपुर-१० (भारत)
V.N.I.T., NAGPUR-440010. (India)

Date: 9/05/2024

विभाग प्रमुख
Head
संगणक विज्ञान एवं अभियांत्रिकी विभाग
Dept. of Computer Science & Engg.
वि.रा.प्रौ.सं. नागपुर-१० (भारत)
V.N.I.T., Nagpur-440010. (India)

ACKNOWLEDGMENTS

We would like to express our sincere gratitude to our guide, Dr. Ashish Tiwari Sir, for guiding us through our project on reducing digital eye strain through continuous screening. His insights and thoughtful direction have been invaluable in addressing a common issue in eye health. We thank him for his consistent support throughout the project duration.us the opportunity of working on a problem that is widely faced by the gadget users.

We would also like to thank Mr. Awasthi sir, M.Tech CSE for his guidance and help in implementing the project and providing the necessary information and resources whenever required by us. We also thank our panel members, Prashanth Kumar sir, Taqi Ali sir, Nidhi Ma'am for providing thoughtful insights in taking the project forward.

Thanking you,

J.C. Manjunath Reddy,
Motamarri Sai Roopak,
Pilli Vishal,
Sangaraju Naga Sathwik.

ABSTRACT

Digital eye strain, also known as computer vision syndrome, is a significant concern in today's technology-driven world, where extended use of gadgets, especially mobile phones, is prevalent. A report by the Vision Council in 2016 noted that, approximately two-thirds of adults aged 30–49 years spend five or more hours on digital devices. The global mobile phone market is projected to grow steadily, with a compound annual growth rate (CAGR) of about 6% from 2021 to 2026. This growth is attributed to the rapid adoption of 5G technology, expanding internet penetration, and increased smartphone use for social media, gaming, and streaming services.

The rampant use of social media is particularly pronounced among younger adults, with reportedly 87% of individuals between 20 and 29 years of age reporting the use of two or more digital devices simultaneously causing digital eye strain in result. Increase in digital eye strain can cause an array of symptoms, including eyestrain, watering of eyes, headache, tired eyes, burning sensation, red eyes, irritation, dry eye, foreign body sensation, blurred vision at near, and double vision.

In an increasingly digitized world, screens are an unavoidable part of daily life. However, with awareness, precautions, and technological advancements, it's possible to protect our vision and navigate the digital age with clear and comfortable eyesight. In today's era, where eye health is of utmost concern due to the widespread use of mobile phones and screens, changing the lifestyle patterns linked to these devices is no easy task. Hence, we used this observation to our advantage. We propose leveraging our extensive mobile phone usage to monitor our eyes actively and craft solutions that help us to avoid digital eye strain.

LIST OF FIGURES

Fig. 1) Facial Landmarks	16
Fig. 2) Eye Aspect Ratio	17
Fig. 3) EAR Formula	17
Fig. 4) Mouth Aspect Ratio (MAR)	18
Fig. 5) Edge detection	20
Fig. 6) Reference method Distance formula	23
Fig. 7) Kivy	25
Fig. 8a) App Component: Buddy App	26
Fig. 8b) App Component: Settings	26
Fig. 9) Workflow	29
Fig. 10) Dlib	30
Fig. 11a) Eye Aspect Ratio terminal result	39
Fig. 11b) Eye Aspect Ratio graph	40
Fig. 12a) Mouth Aspect Ratio terminal result	40
Fig. 12b) Mouth Aspect Ratio graph	40
Fig. 13a) Distance Reference Method terminal	41
Fig. 13b) Distance Reference Method graph	41
Fig. 14a) Diabetic Retinopathy terminal result	42
Fig. 14b) Diabetic Retinopathy train and valid accuracy	42
Fig. 14c) Diabetic Retinopathy train and valid loss	43
Fig. 14d) Diabetic Retinopathy train accuracy at each epoch	44

INDEX

1. Introduction

1.1 Problem Statement	08
1.2 Motivation	08
1.3 Background Challenges	10

2. Literature Survey

2.1 Literature survey of Digital Eye Strain cause	12
2.2 Outlining each problem's Solution	14
2.2.1 Blink Rate and Drowsiness	14
2.2.2 Low Lighting Conditions	15
2.2.3 Viewing at same distance	15
2.2.4 Diabetic Retinopathy Solution	
2.3 Blink Rate	15
2.3.1 Facial Landmarks	15
2.3.2 Eye Aspect Ratio and Mouth Aspect Ratio	17
2.4 Low Lighting Conditions	18
2.4.1 Defining Contrast	18
2.4.2 Edge Detection Algorithms	19
2.5 Viewing at same distance	21
2.5.1 Reference Distance Method	21

3. Design

3.1 Techstack	25
3.2 Workflow of Application	27

4. Implementation

4.1 Landmark Detection model	30
4.2 Methodology of Contrast Analysis	34
4.3 Age Detection model	35

4.4 Diabetic Retinopathy model	36
5. Results and Conclusion	
5.1 Evaluating the architecture performance	39
5.2 Conclusion	44
6. Future Scope	45
List of References	46

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

The prevalence of digital eye strain, exacerbated by factors such as prolonged screen exposure and suboptimal viewing conditions, poses a significant challenge to individuals' visual comfort and productivity. Current solutions often lack comprehensive real-time monitoring and personalized mitigation strategies, while compatibility issues between advanced computer vision libraries and mobile development frameworks further impede the development of effective solutions. Addressing these challenges requires the design and implementation of a multifaceted digital eye strain mitigation system that integrates real-time monitoring, personalized recommendations, and user-friendly interfaces within the applications.

1.2 Motivation

The prevalence of digital eye strain, exacerbated by factors such as prolonged screen exposure and suboptimal viewing conditions, poses a significant challenge to individuals' visual comfort and productivity. Current solutions often lack comprehensive real-time monitoring and personalized mitigation strategies, while compatibility issues between advanced computer vision libraries and mobile development frameworks further impede the development of effective solutions. Addressing these challenges requires the design and implementation of a multifaceted digital eye strain mitigation system that integrates real-time monitoring, personalized recommendations, and user-friendly interfaces within the constraints of mobile application development frameworks.

1.3 Challenges

Developing an effective solution for mitigating digital eye strain presents several challenges. One significant challenge lies in accurately monitoring key factors contributing to eye strain, such as blink rate and drowsiness, in real-time. This requires the development of robust computer vision algorithms capable of processing live video streams and accurately detecting relevant facial features and movements. Additionally, detecting and addressing low lighting conditions poses another challenge.

While ambient lighting plays a crucial role in visual comfort, detecting insufficient lighting conditions and adjusting screen brightness or recommending environmental adjustments require advanced image processing techniques. Furthermore, integrating individual differences in eye characteristics and preferences adds complexity to the system. Factors such as age, pre-existing eye conditions (e.g., dry eyes), and personal habits must be taken into account to provide personalized recommendations and thresholds for digital eye strain mitigation.

An additional challenge encountered during the development process was the compatibility issue between the in-built Dlib and OpenCV libraries, which are primarily written in C++. The Android Native Development Kit (NDK) and Software Development Kit (SDK) versions required for building the Kivy application were found to be incompatible with the versions supported by Dlib and OpenCV. Despite attempts to resolve this compatibility issue, including exploring alternative libraries and adjusting build configurations, a viable solution was not achieved within the scope of this project. As a result, the final implementation of the digital eye strain mitigation system is showcased within the Kivy application framework.

CHAPTER 2

LITERATURE SURVEY

In this chapter, we discuss in detail about causes of Digital Eye Strain. Digital Eye Strain (DES), commonly referred to as computer vision syndrome, has become a prevalent concern in today's digital age. It encompasses a range of discomforts and vision issues that individuals experience during or after prolonged computer use, often involving other digital devices like smartphones and tablets. As society increasingly relies on technology for work, education, and leisure, understanding the causes of DES is paramount to addressing this modern-day health challenge. The causes of DES reveal several contributing factors.

2.1 Literature survey of each problem

Prolonged Screen Exposure

Prolonged screen exposure is a significant factor contributing to Digital Eye Strain (DES). When individuals spend extended periods focusing on digital screens, such as computers, smartphones, and tablets, their eyes are continuously adjusting and refocusing. This constant activity can lead to eye strain, discomfort, and other DES symptoms

The eyes naturally adjust their focus to accommodate different distances, a process called accommodation. When viewing screens, this accommodation is primarily fixed at a single distance, leading to a sustained contraction of the eye muscles. Over time, this can cause fatigue and strain in the eye muscles, resulting in DES symptoms like dryness, irritation, blurred vision and headaches. Moreover, prolonged screen exposure often

involves tasks that require intense visual concentration, such as reading small text or staring at bright screens for extended periods. These activities further exacerbate eye strain and contribute to DES

Poor Ergonomics

In Poor ergonomics, particularly improper viewing distance from the screen, significantly contributes to Digital Eye Strain (DES). Research indicates that maintaining an optimal distance from the screen is crucial in mitigating DES symptoms. Recommended viewing distances typically range from 20 to 26 inches (50 to 65 centimeters) from the eyes to the screen.

When the screen is too close, typically less than 20 inches, the eyes must work harder to focus, leading to increased strain and discomfort. Conversely, if the screen is too far away, beyond 26 inches, users may unconsciously lean forward or squint, further straining the eyes. Therefore, adhering to these recommended distances is essential for minimizing DES symptoms and promoting comfortable and efficient digital device use.

Reduced Blinking Rate

Normal blinking is essential to ensure the normal distribution of the tear film and to protect the ocular surface. Blinking abnormalities may result in poor tear distribution and hence cause damage to the ocular surface. Several studies have investigated the blink rate and the interval between blinks. It has been reported that the normal spontaneous blink rate is between 12 and 15/min. Other studies showed that the interval between blinks ranges from 2.8 to 4 and from 2 to 10 s. A mean blink rate of up to 22 blinks/min has been reported under relaxed conditions. The variability in the blinking measurements of previous studies may be due to differences in experimental conditions.

Viewing in Low lighting conditions

Viewing digital screens in low lighting conditions is a significant contributor to Digital Eye Strain (DES). Research suggests that inadequate lighting can strain the eyes, leading to increased DES symptoms. The contrast between the bright screen and the dim surrounding environment forces the eyes to constantly adjust, resulting in discomfort and fatigue. Moreover, viewing screens in low light can cause the pupils to dilate, allowing more potentially harmful blue light to enter the eyes, exacerbating DES symptoms. Therefore, ensuring proper ambient lighting when using digital devices is essential for reducing eye strain and promoting comfortable screen viewing.

2.2 Outlining each Problem's Solution

2.2.1 Blink Rate and Drowsiness

Solution: If the blink rate falls below a predefined threshold value, trigger a notification to remind the user that blink rate has dropped down and hence to blink more often. Our blink detection task is divided into further sub tasks. In the first subtask, the eye aspect ratio is calculated to determine if a person is blinking or not in a given set of frames. From there, we can proceed to perform following facial landmark detection and blink rate.

2.2.2 Low Lighting conditions

Solution: This problem can also be tackled as the problem to detect low contrast images. A low contrast image has very little difference between light and dark regions, making it hard to see where the boundary of an object begins and the background of the scene starts. Due to poor lighting conditions, the boundaries of the viewer against the background are not well defined — by itself, an edge detection algorithm, such as the Canny edge detector, may struggle to detect the boundary of the object (here the one who is viewing the device).

2.2.3 Viewing at same distance

Solution: For calculating distance between camera and viewer, we can use triangular similarity. Suppose if we have a object with a known width W . We then place this viewer some distance D from our camera. We take a picture of our object using our camera and then measure the apparent width in pixels P . This allows us to derive the perceived focal length F of our camera: $F = (P \times D) / W$ For this calculation we need to set the initial.

2.2.4 Diabetic Retinopathy solution

Solution: We've developed a model that utilizes Fundoscopy to capture images of the eye, aided by a 20-D diopter lens attached to a camera. These images can then be uploaded to the Kivy platform for analysis, providing corresponding results. Our model is trained to predict various stages of retinopathy, specifically focusing on diabetic retinopathy

2.3 Blink Rate

2.3.1 Facial Landmarks

Facial landmarks are specific points on the face that are anatomically significant for various applications, including facial recognition, medical imaging, and computer graphics. These landmarks serve as reference points for analyzing and identifying facial features and expressions. Common facial landmarks include the corners of the eyes, nose tip, mouth corners, and chin point. They are used in algorithms and techniques for tasks such as face detection and facial feature tracking.

We used the famous dlib's Shape predictor in phase 1 and later we trained our own one. dlib's 68 landmarks detector is a popular facial landmark detection algorithm used in computer vision applications. It provides a set of 68 key points that represent various facial features, including the eyes, nose, mouth, and chin. These landmarks are detected using a combination of shape prediction models and machine learning techniques. For the eyes, dlib's 68 landmarks detector typically identifies several key points:

Left Eye Landmarks:

Inner Eye Corner (Left): Landmarks 36 to 41

Outer Eye Corner (Left): Landmarks 42 to 47

Top Eyelid (Left): Landmarks 48 to 54

Bottom Eyelid (Left): Landmarks 55 to 59

Right Eye Landmarks:

Inner Eye Corner (Right): Landmarks 42 to 47

Outer Eye Corner (Right): Landmarks 36 to 41

Top Eyelid (Right): Landmarks 48 to 54

Bottom Eyelid (Right): Landmarks 55 to 59

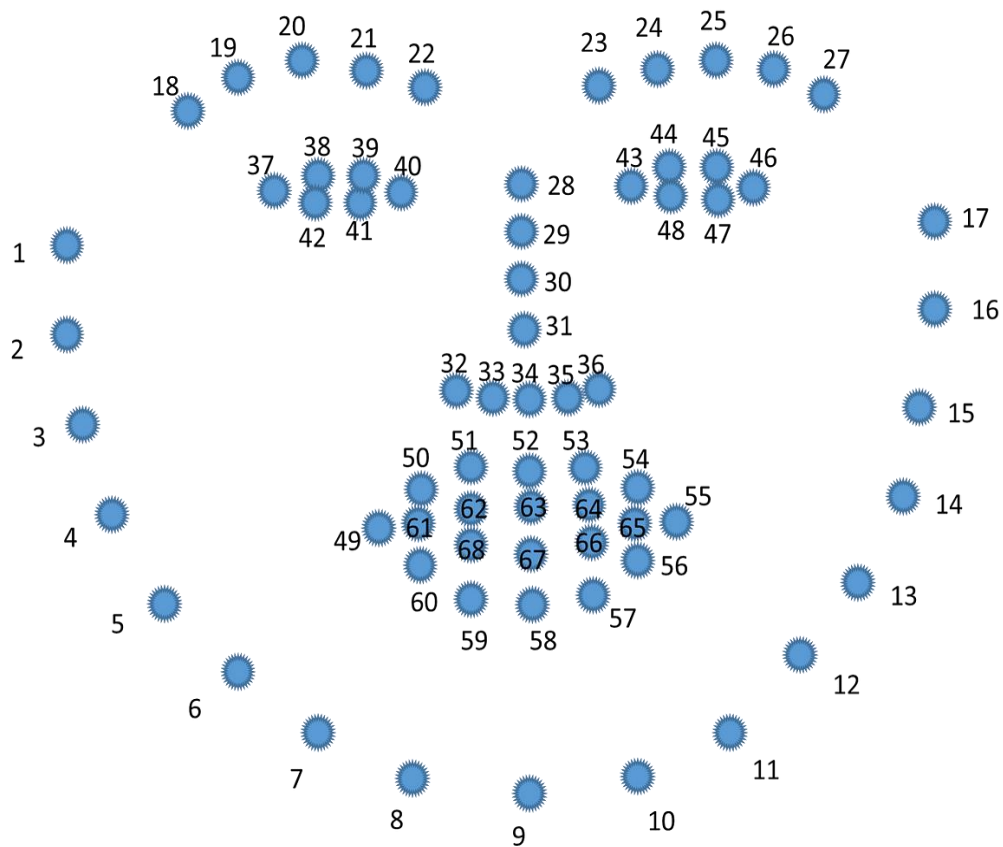


Fig 1

2.3.2 Eye Aspect Ratio and Mouth Aspect Ratio

we can apply facial landmark detection to localize important regions of the face, including eyes, eyebrows, nose, ears, and mouth. In terms of blink detection, we are only interested in two sets of facial structures — the eyes. Each eye is represented by 6 (x, y)-coordinates, starting at the left-corner of the eye (as if you were looking at the person), and then working clockwise around the remainder of the region:

There is a relation between the width and the height of these coordinates. Based on the work by Soukupová and Čech in their 2016 paper, Real-Time Eye Blink Detection using Facial Landmarks, we can then derive an equation that reflects this relation called the eye aspect ratio.

Similarly, we derive for mouth called mouth aspect ratio for detecting yawn for drowsiness detection.

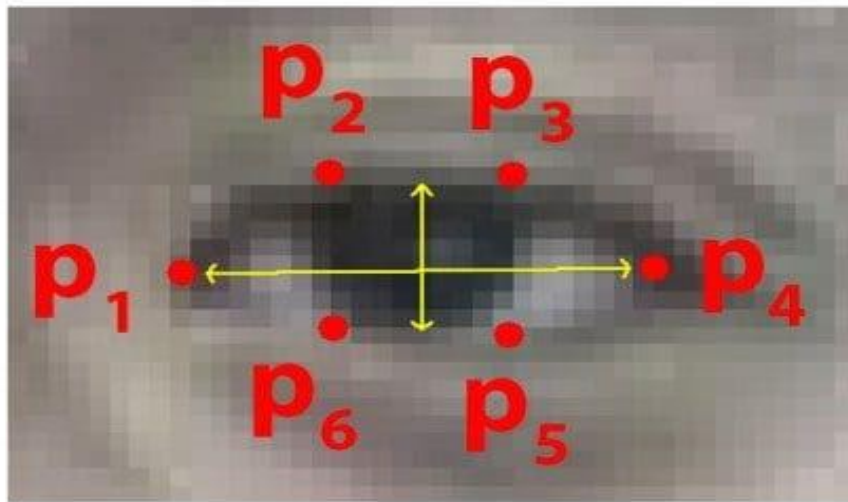
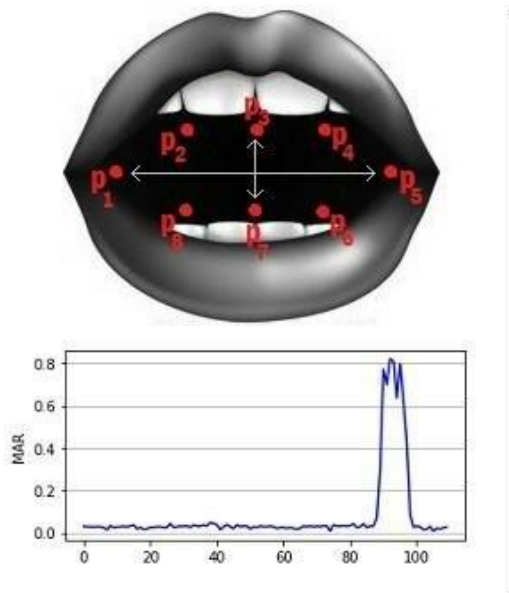


Fig 2

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

Fig 3



$$\text{MAR} = \frac{\|p_2 - p_8\| + \|p_3 - p_7\| + \|p_4 - p_6\|}{2\|p_1 - p_5\|}$$

Fig 4

2.4 Low Lightning Conditions

2.4.1 Contrast

Image contrast refers to the difference in intensity between the light and dark areas of an image. It determines the visibility and distinction of objects or features in the image. High contrast images have strong differences between light and dark areas, making objects stand out more distinctly, while low contrast images have minimal differences, resulting in a more subdued appearance.

In a well-contrasted image, edges are sharply defined and distinguishable. This means that there is a clear and noticeable difference in intensity or color between adjacent pixels along the edges of objects or features in the image. These strong variations in intensity or color allow for precise localization and delineation of edges, contributing to enhanced clarity and visual perception of the image content.

So, we bundled down the problem as the problem of differentiating the frame boundaries with the intensity values.

2.4.2 Edge Detection Algorithm

An edge in an image is a significant local change in the image intensity. As the name suggests, edge detection is the process of detecting the edges in an image. Discontinuities in depth, surface orientation, scene illumination variations, and material properties changes lead to discontinuities in image brightness. Thus, applying an edge detection algorithm to an image may significantly reduce the amount of data to be processed. This therefore filters out information that may be regarded as less relevant while preserving the important structural properties of an image.

There are multiple approaches for edge detection.

1. Traditional Approach
2. Deep Learning Approach

Traditional Approaches include Robert, Sober, Prewitt filters and Deep Learning Approaches include Canny Edge detector algorithm and Holistically Nested Edge Detector algorithm. We have chosen Canny Edge Algorithm

The following are the steps involved in Canny Edge Detection:

1. Image Smoothing

Edge detection is sensitive to noises in image. Noises in image are unwanted variations of pixel values that do not represent the actual sense. The first step is to remove noise with help of gaussian filters.

2. Gradient Calculation

We obtained a smooth image and now we have to find the edge strength by taking gradient of image. Sobel operators uses a 2D spatial gradient operator measurement on image.

3. Non-Maximum Suppression

We desire to obtain thin edges and hence decide to perform non maximum suppression to thin out the edges. The principle is simple: the algorithm goes through all the points on the gradient intensity matrix and finds the pixels with the maximum value in the edge directions

4. Hysteresis

In thresholding, we obtain two threshold values say T_{min} and T_{max} .

Whatever the edges with intensity gradient value greater than T_{max} are considered as edges and those of less than T_{min} are grouped as non-edges.

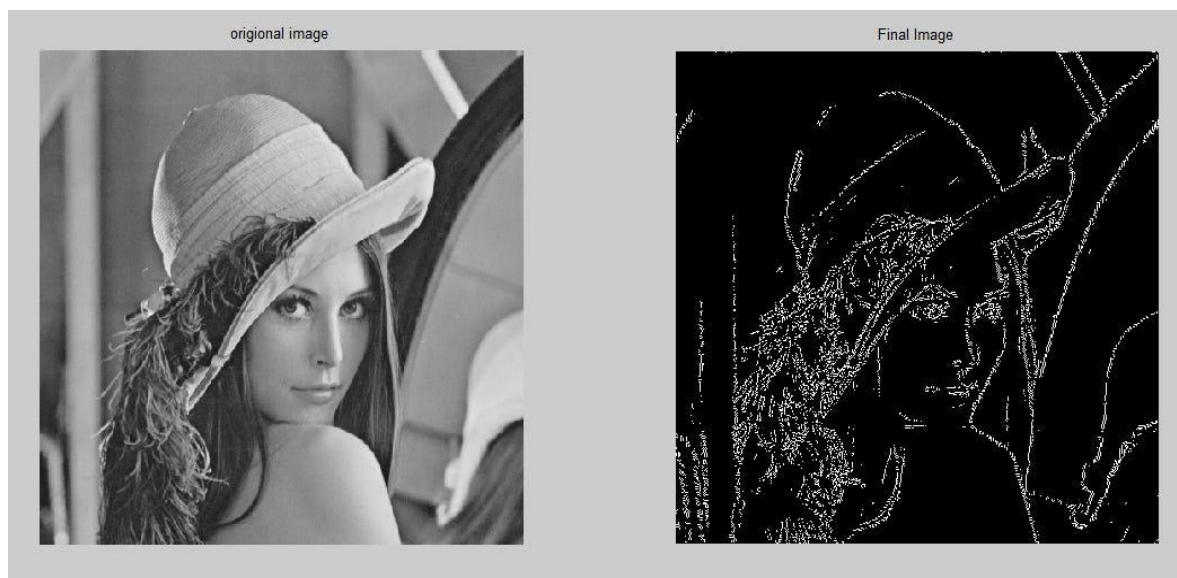


Fig 5

2.5 Viewing at same Distance

2.5.1 Reference Distance Method

Digital Screen Usage: In scenarios where a camera is integrated with digital screens or displays (e.g., video conferencing, interactive kiosks), evaluating the distance of the user's face from the camera becomes crucial. Maintaining an optimal distance ensures better visibility and interaction with the screen. If the user is too close or too far from the screen, it can affect readability, clarity, and the overall user experience. By evaluating the distance and providing feedback, users can adjust their position for optimal viewing and interaction.

Eye Health: Prolonged exposure to digital screens at incorrect distances can strain the eyes, leading to discomfort, fatigue, and potentially long-term vision problems. By notifying users to adjust their distance from the screen, we promote healthy screen usage habits and reduce the risk of eye-related issues.

Accident Prevention: In public settings or interactive installations, maintaining a safe distance from digital displays is essential to prevent accidents or collisions. By alerting users when they are too close to the screen, we mitigate the risk of unintended physical contact or injury.

Adaptation to User Preferences: Notifying users about their distance from the camera allows for personalized adjustments based on individual preferences or requirements. Some users may prefer to sit closer or farther from the screen based on their visual acuity, comfort level, or privacy concerns. Providing distance feedback empowers users to tailor their experience according to their preferences.

How code works?

Define constants:

KNOWN_DISTANCE: The actual distance (in centimeters) between the camera and the object (face), which is known beforehand.

KNOWN_WIDTH: The actual width (in centimeters) of the object (face) being measured.

Set up colors and fonts for displaying text on the output frame.

Initialize the video capture object (cap) to capture video frames from the camera.

Load the pre-trained face detector cascade classifier using CascadeClassifier.

Focal Length Calculation (focal_length function):

Given the known distance (KNOWN_DISTANCE), the real width of the object (KNOWN_WIDTH), and the width of the object in the reference frame image (ref_image_face_width), calculate the focal length of the camera.

The formula used is: $\text{focal_length} = (\text{width_in_rf_image} * \text{measured_distance}) / \text{real_width}$.

Face Detection and Width Measurement (face_data function):

- Convert the captured frame to grayscale for faster processing.
- Use the pre-trained face detector (face_detector) to detect faces in the frame.
- For each detected face:
 - Draw a rectangle around the face on the frame.
 - Measure the width of the face.

Distance Calculation (distance_finder function):

Given the focal length (focal_length_found), the known width of the object (KNOWN_WIDTH), and the width of the face in the frame (face_width_in_frame), calculate the distance of the face from the camera.

Display the captured frame with text indicating the calculated distance (in centimeters) of the face from the camera.

Optionally, check if the distance remains consistent over time. If not, suggest changing the distance for better accuracy.

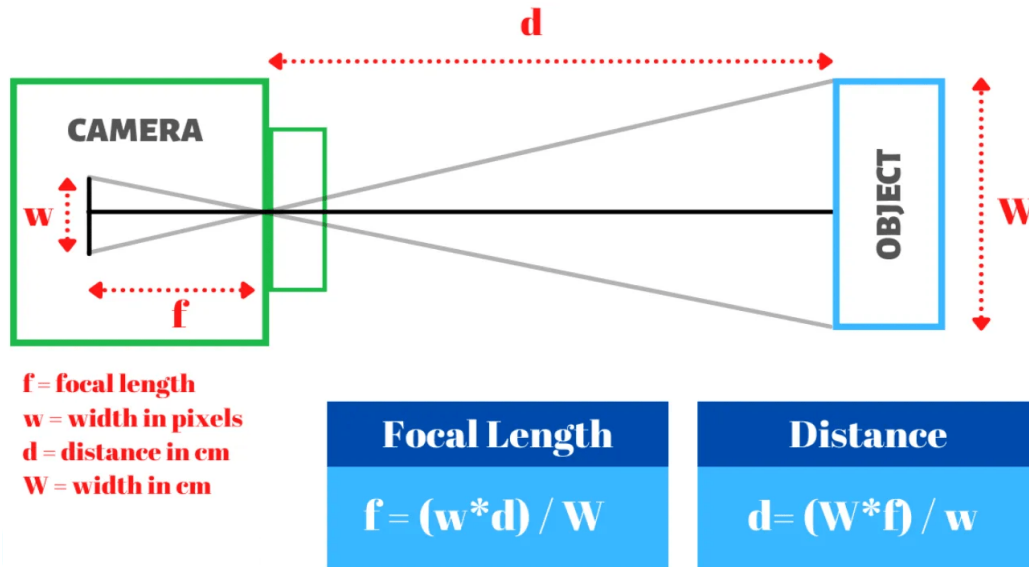


Fig 6

Technical Aspects:

Face Detection: Haar Cascade Classifier is utilized for face detection, which is a machine learning-based approach where a cascade function is trained from a lot of positive and negative images to detect objects.

Focal Length Calculation: The focal length of the camera is calculated using the known distance and width of the object in the reference image, along with the width of the object in the reference image.

Distance Calculation: Once the focal length is determined, the distance to the face is calculated using the real width of the object and the width of the object in the frame.

Consistency Check: The code includes a mechanism to alert the user if the distance estimation remains consistent for a certain duration, suggesting a change in distance if needed for improved accuracy.

CHAPTER 3

DESIGN

In this chapter, we shall discuss in detail about design and workflow of application.

We plan to create an interface that would show the results along with OPENCV video interface from selfie camera through which the camera captures happens and the images are then evaluated.

3.1 Techstack

Kivy for Frontend

Kivy is an open-source Python framework for developing multi-touch applications. It enables developers to create cross-platform applications that can run on Windows, macOS, Linux, Android, and iOS. Kivy is particularly well-suited for creating user interfaces (UIs) with touch capabilities, making it ideal for mobile and tablet applications.

The KV design language, often referred to as kvlang or the Kivy language, is a core component of the Kivy framework. It's a declarative language specifically designed to streamline the creation of user interfaces (UIs) for Kivy applications.



Fig

Components:

• BuddyApp:

Integrates various detection classes into a single application.

Uses threading to continuously update the frame from the video capture without Block. In the main UI thread, provides a GUI for real-time feedback on detected metrics.

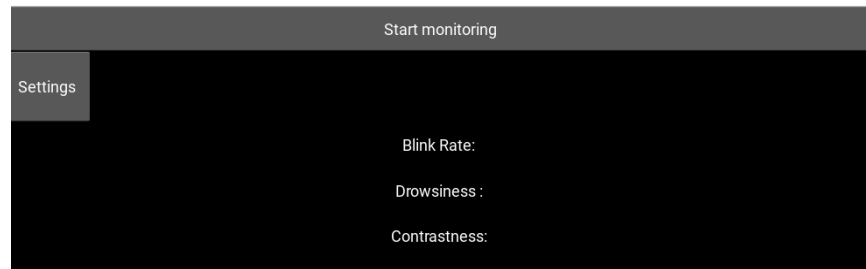


Fig 8a

• SettingsPopup:

A popup window allowing users to adjust application settings, such as the contrast threshold.

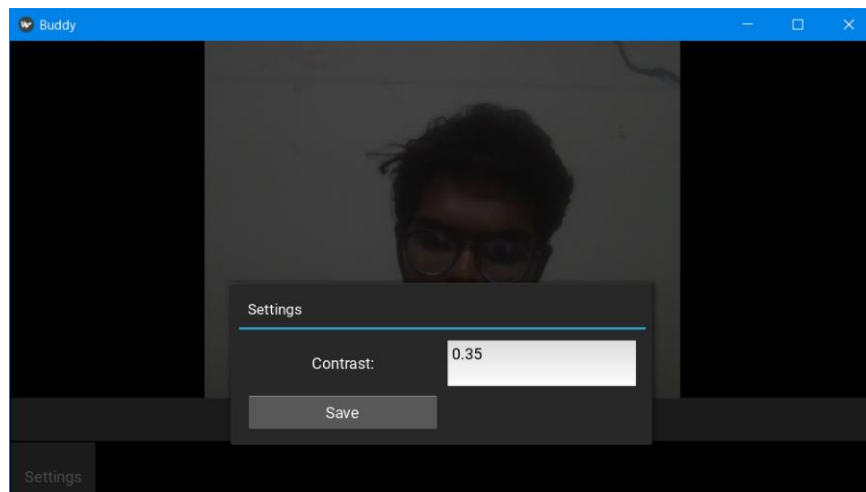


Fig 8b

Key Features:

Real-Time Video Processing: Utilizes OpenCV for capturing and processing video frames in real-time.

Multithreaded Frame Update: Employs a separate thread to update frames from the video capture, ensuring the UI remains responsive.

Modular Detection Classes: Incorporates separate classes for different monitoring tasks, demonstrating a modular approach to system design.

User-adjustable Settings: Features a settings popup for dynamic adjustment of application parameters naming image contrast threshold.

3.1 Workflow

1. Initialization:

- The application imports necessary libraries for Kivy UI, OpenCV computer vision, threading, and custom modules for blink detection, drowsiness detection, contrast detection, and distance measurement.
- It creates a VideoCapture object to access the webcam.
- A SettingsPopup class is defined to allow users to adjust contrast thresholds.
- Multithreading components are set up for frame updates and processing.
- Objects are created for each detection/measurement module (blink detection, drowsiness detection, contrast detection, distance measurement).

2. Building the User Interface (UI):

- The build method constructs the UI layout using a Box Layout.
- The layout includes:
 - An Image widget to display the webcam feed.
 - A Button to start monitoring.
 - A Button to open the settings popup.
- Several Label widgets to display the results (blink rate, drowsiness, contrast level, distance).

3. User Interaction:

- Clicking the "Settings" button opens the SettingsPopup, allowing users to modify the contrast threshold.
- Clicking the "Start monitoring" button initiates the frame processing loop.

4. Frame Processing Loop:

- A separate thread is started to continuously capture frames from the webcam.
- Within this loop:
- A new frame is captured and stored with a lock to prevent race conditions.
- The *process_frame* method is scheduled to run every 1/10th of a second (10 frames per second).

5. Frame Processing:

- The *process_frame* method retrieves the current frame with thread-safe access.
- If a frame is available:
- The UI is updated with the latest webcam feed (flipped horizontally for a mirrored view).
- Each measurement module is called to process the frame:
- *blink_detection.process_blink_detection(frame)*: Analyzes the frame for blinks and updates internal counters.
- *drowsiness_detection.process_yawn_detection(frame)*: Analyzes the frame for signs of drowsiness and updates an internal state.
- *contrast_detection.detect_contrast(frame, contrast)*: Analyzes the frame's contrast based on the user-defined threshold and updates internal values.
- *distance_measurement.detect_distance(frame)*: Performs distance measurement using a potentially user-configurable method.

The UI labels are updated with the results from each module.

Overall, the BuddyApp follows a continuous loop:

- Capture a frame from the webcam.
- Update the UI with the latest frame.
- Process the frame for various wellness factors using separate modules.
- Update the UI labels with the detection/measurement results.
- This report articulation effectively highlights the application's functionality and the interaction between its components.

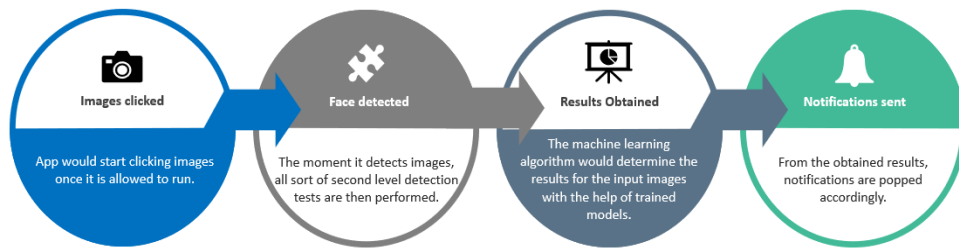


Fig 9

CHAPTER 4

IMPLEMENTATION

In this chapter, we shall discuss in detail about the implementation of architectures used to process the image and extract the needed features accordingly.

4.1 Landmark Detection Model Architecture

Initially we used dlib's 68 facial landmarks detector to carry out the image processing and later trained one with 106 custom facial landmark detection model using *MobileNetV2*.



Fig 10

Dlib

Dlib is a powerful and comprehensive open-source C++ library designed to empower developers in creating real-world applications that leverage machine learning and data analysis techniques. It boasts a vast array of tools and algorithms, making it a valuable asset for various domains, including Machine Learning, Data Analysis, Image Processing.

OPENCV

OpenCV (Open-Source Computer Vision Library) is a free and open-source library that serves as a cornerstone for real-time computer vision applications. It boasts a massive collection of algorithms (over 2500!) and caters to various programming languages, including:

- C++ (primary language)
- Python (popular choice for ease of use)
- Java
- C
- Assembly language

This versatility makes OpenCV a valuable asset for developers working across different environments. Here's a breakdown of OpenCV's strengths:

Core Functionalities:

Image and Video Processing: OpenCV excels at manipulating and analyzing images and videos. It provides tools for:

- Image filtering (noise reduction, sharpening, etc.)
- Feature extraction (identifying key characteristics within images)
- Image segmentation (dividing images into meaningful regions)
- Motion analysis in videos

Object Detection and Recognition: OpenCV empowers you to detect and recognize objects within images and videos. This includes:

- Identifying specific objects (e.g., cars, faces)
- Tracking object movement
- Counting objects in a scene

Machine Learning Integration: OpenCV seamlessly integrates with machine learning frameworks, allowing you to build intelligent systems:

- Train custom object detectors using machine learning techniques
- Leverage pre-trained models for tasks like facial recognition
- Real-time Processing: OpenCV is optimized for real-time applications, making it suitable for:
- Video surveillance systems

MobileNetV2 is a convolutional neural network (CNN) architecture specifically designed for mobile and embedded devices. It excels at striking a balance between model size and accuracy, making it ideal for applications where computational resources are limited. Here's a breakdown of its key components:

Building Blocks:

Inverted Residual Blocks: Unlike traditional residual blocks, MobileNetV2 uses inverted residuals. The residual connection is placed between bottleneck layers, allowing for efficient feature extraction.

Depth wise Separable Convolutions: These convolutions decompose a standard convolution into two separate steps:

Depth wise convolution: Applies a single filter to each input channel, extracting features efficiently.

Pointwise convolution: Uses 1x1 filters to combine the features from the depth wise step, reducing computational cost.

Linear Bottlenecks: These layers introduce a lower-dimensional representation within the inverted residual blocks, further reducing model size without sacrificing accuracy.

Squeeze-and-Excitation (SE) Blocks (Optional): These blocks can be incorporated to adaptively recalibrate feature channels, potentially improving model performance.

Overall Architecture:

Initial Convolution: A standard convolution layer processes the input image.

Inverted Residual Blocks (19x): These blocks, with varying configurations, extract features at different resolutions.

Final Convolution: A final convolution layer generates the model's output, typically for image classification tasks.

Dataset for Landmark Detection

Lapa

Key Characteristics of LaPa-Dataset:

- **Large Scale:** LaPa offers a substantial dataset of images, typically exceeding 22,000 images. This vast amount of data helps train robust and generalizable models.
- **Rich Annotations:** Each image in LaPa comes with two types of annotations:
- **Pixel-level Labels:** Each pixel in the image is assigned a label corresponding to a specific facial region (typically 11 categories are used, including background). This enables pixel-wise segmentation of facial parts.
- **Facial Landmarks:** Each image also includes the location of 106 key points on the face (e.g., corners of eyes, tip of nose). These landmarks provide valuable information about facial structure and pose.
- **Variations:** The images in LaPa encompass a diverse range of facial features, including:
 - Expressions (smiling, frowning, neutral)
 - Poses (frontal, profile, tilted)
 - Occlusions (glasses, hats, hair)
 - Ethnicity and age variations

The LaPa-Dataset is typically divided into training and testing sets to facilitate the training and evaluation of models. The split follows the structure:

Training Set: The majority of the images (around 24,000 or 21,866) are allocated to the training set. This large training dataset allows models to learn effectively from a wide variety of examples.

Testing Set: A smaller portion of the images (around 2,800 or 1,000) is reserved for the testing set. This set is used to assess the performance of trained models on unseen data.

Now that we successfully obtained the facial landmarks, we can then obtain landmarks of eye and then calculate Eye Aspect Ratio and Mouth aspect Ratio of frame.

Blink Rate and Yarn Detection

For a set of frames, both aspect ratios of all frames are calculated and then average of them are then compared with threshold aspect ratios (experimentally)

We considered the following threshold aspect ratios from studies [1]

Threshold Eye Aspect Ratio for Blink Detection: 0.3

Threshold Mouth Aspect Ratio for Yarn Detection: 0.6

4.2 Methodology of Contrast Analysis

The method is to first convert the frame to grayscale.

Then utilize the *is_low_contrast* function from the *skimage.exposure* library to assess the grayscale image's contrast level based on a user-defined threshold.

Based on the contrast level, a text string ("Low contrast" or "Better contrast") is stored in the *contrast_text* attribute of the class.

Edge Detection:

- A Gaussian blur is applied to the grayscale image for noise reduction.
- The Canny edge detection algorithm is then used to identify edges within the blurred image.

Lighting Condition Determination:

- The method calculates the percentage of pixels classified as edges by dividing the number of non-zero pixels in the edge image by the total number of pixels in the image and multiplying by 100. This value is stored in the *edge_percentage* attribute.
- It compares the *edge_percentage* with a user-defined *contrast_percentage* value.

- If the *edge_percentage* is lower than the *contrast_percentage*, the lighting condition is considered "*Bad Lighting*". Otherwise, the lighting condition is considered "*Good Lighting*".

The lighting condition is appended to the *contrast_text* attribute to communicate to the frontend interface.

4.3 Age Estimation Model Architecture

Our approach involved architecting a sequential model comprising convolutional and dense layers, strategically configured to extract hierarchical features from input images and infer age groups. Through a systematic design process, we aimed to optimize the network's capacity to learn discriminative patterns inherent in facial structures indicative of age. By incorporating convolutional layers with increasing filter sizes and leveraging techniques such as average pooling and global average pooling, our model was engineered to capture both local and global features essential for accurate age classification.

Input Layer: The network begins with an input layer that accepts grayscale images of size 200x200 pixels.

Convolutional Layers: There are four convolutional layers, each followed by an average pooling layer. The first convolutional layer has 32 filters, while subsequent layers have 64, 128, and 256 filters respectively. Each convolutional layer uses a 3x3 kernel and ReLU activation function.

Pooling Layers: After each convolutional layer, there is an average pooling layer with a 2x2 pool size, which helps in reducing the spatial dimensions of the feature maps.

Global Average Pooling Layer: Following the last convolutional layer, a global average pooling layer is applied. This layer computes the average value of each feature map, resulting in a single value for each filter. This helps in reducing the spatial dimensions of the feature maps to a single value per filter.

Dense Layers: After the global average pooling layer, there is a dense layer with 132 nodes, followed by an output layer with 7 nodes. The dense layer uses ReLU activation

function, while the output layer uses softmax activation function. The output layer has 7 nodes, corresponding to the number of classes in the classification task.

4.4 Diabetic Retinopathy Detection Architecture

VGG16 Architecture:

The VGG16 (Visual Geometry Group 16) architecture is a deep convolutional neural network (CNN) introduced by the Visual Geometry Group. It gained popularity for its simplicity and effectiveness in image classification tasks. The architecture comprises 16 layers, including 13 convolutional layers and 3 fully connected layers.

Convolutional Layers:

1. The network starts with a series of convolutional layers, where each layer applies a set of learnable filters to the input image, extracting features at different spatial scales.
2. The convolutional layers use small receptive fields (typically 3x3) with a stride of 1 and zero-padding to maintain the spatial dimensions of the input.
3. After each convolutional layer, a rectified linear unit (ReLU) activation function is applied to introduce non-linearity into the network.

Max Pooling Layers:

1. Max pooling layers are inserted after certain convolutional layers to down sample the feature maps and reduce computational complexity.
2. Max pooling operates by selecting the maximum value within each pooling window, effectively reducing the spatial dimensions of the feature maps while retaining important information.

Fully Connected Layers:

1. After several layers of convolution and pooling, the feature maps are flattened and fed into fully connected layers.

2. These layers act as classifiers, combining the extracted features to make predictions about the input image's class labels.
3. The last fully connected layer typically employs a softmax activation function to produce probability scores for each class, indicating the likelihood of the input image belonging to a particular category.

Transfer Learning:

Transfer learning is a machine learning technique where knowledge gained from training one model is transferred or reused to improve the performance of another related task or model. In the context of deep learning, transfer learning involves using pre-trained models, such as VGG16, that have been trained on large datasets (e.g., ImageNet) to extract meaningful features from images. By leveraging the pre-trained model's learned representations, transfer learning allows us to overcome the limitations of training deep networks from scratch, particularly when working with limited training data.

Pre-trained Model Usage:

- In transfer learning, we use the pre-trained VGG16 model as a feature extractor. The convolutional layers of VGG16 have learned to extract meaningful hierarchical features from images during training on ImageNet.
- Instead of training the entire network from scratch, we freeze the weights of the pre-trained layers to retain the learned representations and prevent them from being modified during training.

Customization for Specific Task:

- We replace the original fully connected layers of VGG16 with new layers customized for our specific task, such as age estimation or diabetic retinopathy detection.
- These new layers are randomly initialized and trained using the features extracted by the pre-trained convolutional layers.

Fine-tuning:

- To adapt the model to our target task, we fine-tune the weights of the entire network, including both the pre-trained convolutional layers and the newly added layers.
- During fine-tuning, we adjust the network's parameters using a smaller dataset specific to our task, allowing the model to learn task-specific patterns while retaining the general features learned from the pre-trained model.

In our approach, we employ transfer learning by using the pre-trained VGG16 model as a feature extractor. We remove the last fully connected layers of VGG16 and replace them with new layers customized for our specific task, such as diabetic retinopathy detection. By fine-tuning the weights of the remaining layers and training only the newly added layers on our dataset, we can efficiently adapt the pre-trained model to our target task, achieving better performance with less computational resources and training data.

CHAPTER 5

RESULTS AND CONCLUSION

In the previous chapter, we have already seen how the problem was broken down from literature survey to implementation. Now in this chapter let's see how we evaluated work.

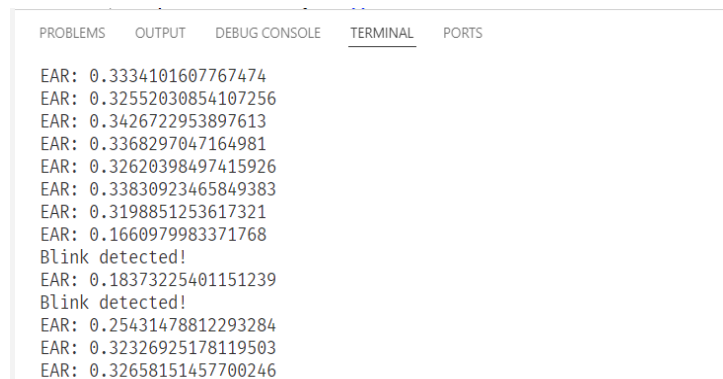
5.1 Evaluating the network performance

To see how good our solutions working we tested our methods with different possible scenarios with respect to the problem. We generated the graphs of respective method we described so far in this report

Facial Landmark Detection Results

In analyzing the effectiveness of our solutions, we focused on two key metrics: eye aspect ratio and mouth aspect ratio. By examining these ratios across various scenarios, we gain insight into how well our methods address the problem at hand. Here are the key findings regarding eye and mouth aspect ratios:

Eye Aspect Ratio



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

EAR: 0.3334101607767474
EAR: 0.32552030854107256
EAR: 0.3426722953897613
EAR: 0.3368297047164981
EAR: 0.32620398497415926
EAR: 0.33830923465849383
EAR: 0.3198851253617321
EAR: 0.1660979983371768
Blink detected!
EAR: 0.18373225401151239
Blink detected!
EAR: 0.25431478812293284
EAR: 0.32326925178119503
EAR: 0.32658151457700246
```

Fig 11a

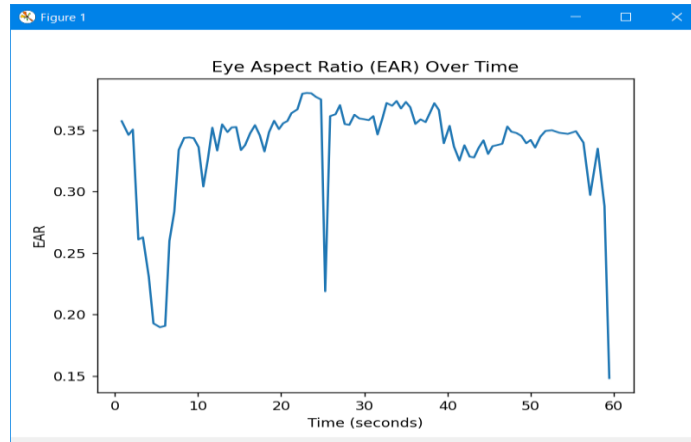


Fig 11b

Mouth Aspect Ratio

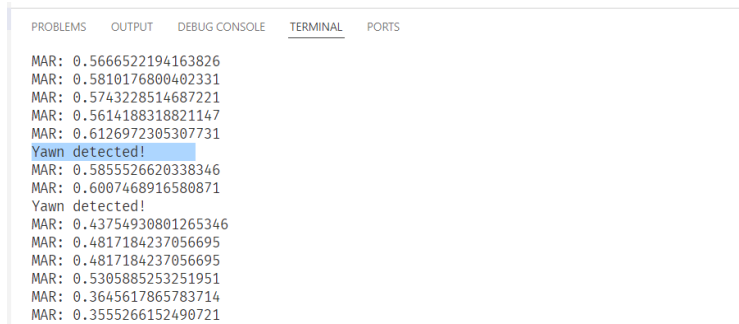


Fig 12a

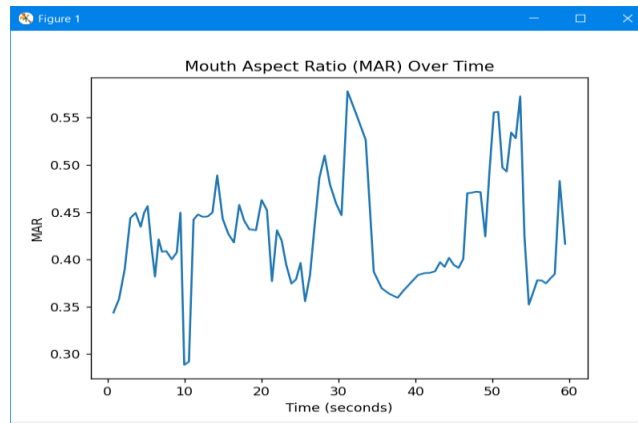


Fig 12b

Distance Reference method

```
Distance: 48.71 CM
Consider changing your distance for better accuracy!
Distance: 48.4 CM
Consider changing your distance for better accuracy!
Distance: 48.4 CM
Consider changing your distance for better accuracy!
Distance: 48.4 CM
Consider changing your distance for better accuracy!
Distance: 47.8 CM
Distance: 46.64 CM
Distance: 46.64 CM
Distance: 46.93 CM
Distance: 45.27 CM
Distance: 45.01 CM
Distance: 44.23 CM
Distance: 43.98 CM
Distance: 43.24 CM
Distance: 43.48 CM
```

Fig 13a

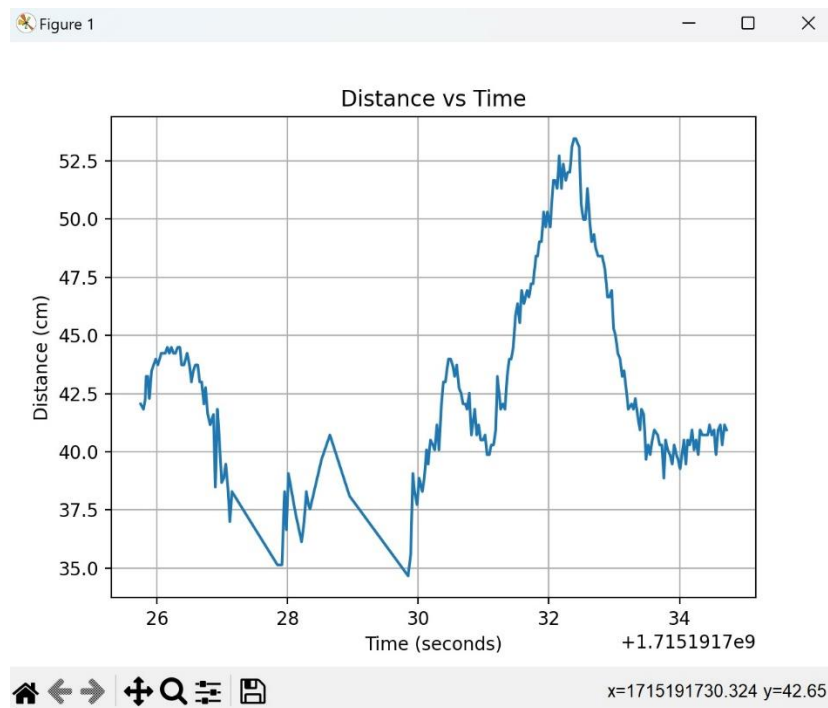


Fig 13b

Diabetic Retinopathy

```
Epoch 15/20
69/69 [=====] - 188s 3s/step - loss: 1.0444 - accuracy: 0.6336 - val_loss: 1.1896 - val_accuracy: 0.6509
Epoch 16/20
69/69 [=====] - 222s 3s/step - loss: 0.9460 - accuracy: 0.6473 - val_loss: 1.0046 - val_accuracy: 0.6636
Epoch 17/20
69/69 [=====] - 196s 3s/step - loss: 0.9827 - accuracy: 0.6505 - val_loss: 1.1746 - val_accuracy: 0.6091
Epoch 18/20
69/69 [=====] - 184s 3s/step - loss: 0.8784 - accuracy: 0.6718 - val_loss: 0.9482 - val_accuracy: 0.6509
Epoch 19/20
69/69 [=====] - 188s 3s/step - loss: 0.9325 - accuracy: 0.6473 - val_loss: 1.0257 - val_accuracy: 0.6291
Epoch 20/20
69/69 [=====] - 189s 3s/step - loss: 0.9647 - accuracy: 0.6505 - val_loss: 0.9928 - val_accuracy: 0.6782
C:\Users\sairo\.conda\envs\dl\Lib\site-packages\keras\src\engine\training.py:3000: UserWarning: You are saving your model as an HDF5 file via
`model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras'
)`).
  saving_api.save_model(
PS C:\Users\sairo\OneDrive\Desktop\deep_learning\retinopathy> █
```

Fig 14a

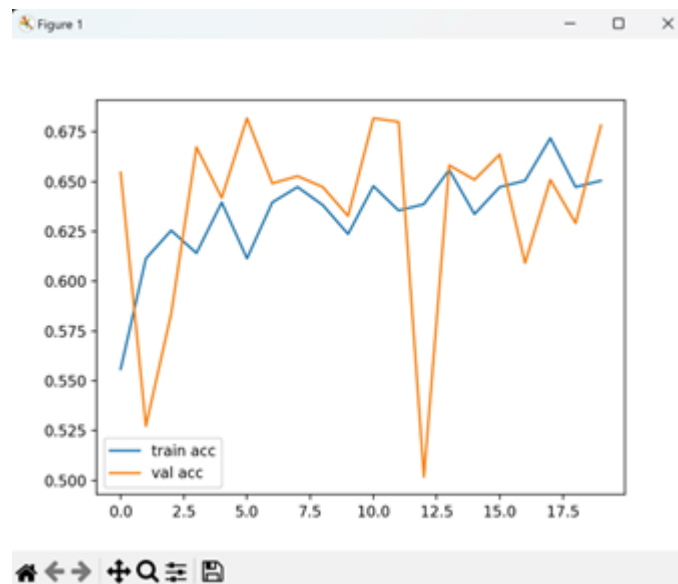


Fig 14b

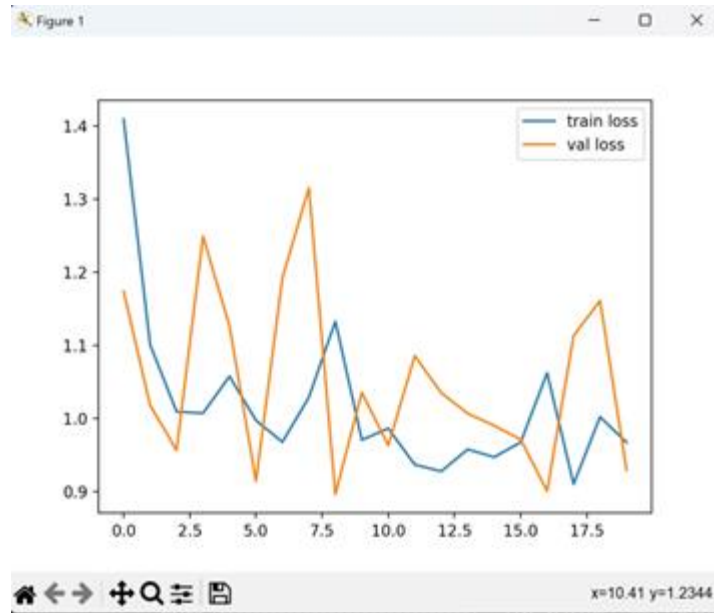


Fig 14c

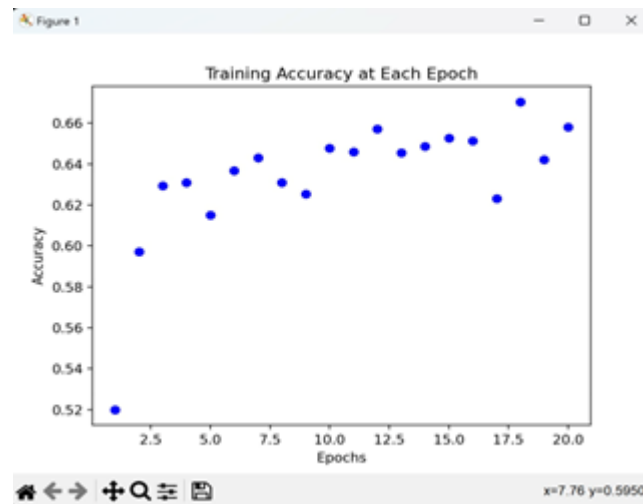


Fig 14d

5.2 Conclusion

In this project, our primary objective revolved around addressing the issues contributing to digital eye strain. As we delved into identifying these problems, we pinpointed several key areas that demanded our attention. Subsequently, we initiated our efforts to tackle these areas, eventually leveraging image processing techniques to gain insights into the condition of the eyes through continuous monitoring.

Throughout this journey, we familiarized ourselves with various technologies related to image processing. Notably, we explored libraries like dlib, which specialize in addressing scenarios requiring precise eye movement analysis. This exploration allowed us to grasp the significance of developing applications that comprehensively detail the status of the eyes, aiding in the prevention of digital eye strain.

Acknowledging the valuable insights provided by existing research, particularly in calculating aspect ratios and estimating distances through facial landmarks, we proceeded to implement practical solutions to mitigate real-world challenges.

By integrating these techniques and methodologies, we aimed to create effective solutions that not only address the root causes of digital eye strain but also empower users to maintain healthy eye habits in the digital age.

CHAPTER 6

FUTURE SCOPE

The future scope of this project is vast, with several avenues for further development and enhancement to create a more robust and effective solution for combating digital eye strain.

Building a Complete Application: Transitioning from a local machine setup to a mobile application can significantly broaden the accessibility and usability of the solution. Integrating the frontend with the backend infrastructure will streamline data processing and enhance the overall user experience.

Improving Model Robustness: While the current models for landmark detection, age estimation, and diabetic retinopathy detection form a solid foundation, there is room for improvement in their robustness and accuracy. By incorporating additional features and data points, such as contextual information, environmental factors, and user-specific characteristics, the models can be fine-tuned to deliver more precise and reliable results.

Expanding Feature Set: Beyond the existing features, there are numerous avenues for expanding the functionality of the application. For instance, integrating additional models or algorithms for detecting drowsiness, eye fatigue, or other indicators of eye strain can provide users with comprehensive insights into their eye health.

Overall, by addressing these areas of improvement, the project can evolve into a comprehensive and cutting-edge solution for alleviating digital eye strain and promoting long-term eye health in the digital era.

REFERENCES

List of references used for this project

[1] EAR and BlinkRate details

Al-Gawwam & Benaissa (2018) Al-Gawwam S, Benaissa M. Robust eye blink detection based on eye landmarks and Savitzky-Golay filtering. Information. 2018;9(4):93–104. doi: 10.3390/info9040093.

[2] D. Torricelli, M. Goffredo, S. Conforto, and M. Schmid. An adaptive blink detector to initialize and update a view-based remote eye gaze tracking system in a natural scenario. Pattern Recogn. Lett., 30(12):1144–1150, Sept. 2009.

[3] X. Xiong and F. De la Torre. Supervised descent methods and its applications to face alignment. In Proc. CVPR, 2013. 2, 3, 4, 5 Z. Yan, L. Hu, H. Chen, and F. Lu. Computer vision syndrome: A widely spreading but largely unknown epidemic among computer users. Computers in Human Behaviour, (24):2026–2042, 2008.

[4] F. Yang, X. Yu, J. Huang, P. Yang, and D. Metaxas. Robust eyelid tracking for fatigue detection. In ICIP, 2012.

[5] S. Zafeiriou, G. Tzimiropoulos, and M. Pantic. The 300 videos in the wild (300-VW) facial landmark tracking in-the-wild challenge. In ICCV Workshop, 2015. <http://ibug.doc.ic.ac.uk/resources/300-VW/>.

[6] Cai J, Boulton M. The pathogenesis of diabetic retinopathy: old concepts and new questions. Eye (Lond). 2002 May;16(3):242-60. [[PubMed](#)]

[7] Mori K, Duh E, Gehlbach P, Ando A, Takahashi K, Pearlman J, Mori K, Yang HS, Zack DJ, Etyreddy D, Brough DE, Wei LL, Campochiaro PA. Pigment epithelium-derived factor inhibits retinal and choroidal neovascularization. J Cell Physiol. 2001 Aug;188(2):253-63. [[PubMed](#)]

- [8] Freyberger H, Bröcker M, Yakut H, Hammer J, Effert R, Schifferdecker E, Schatz H, Derwahl M. Increased levels of platelet-derived growth factor in vitreous fluid of patients with proliferative diabetic retinopathy. *Exp Clin Endocrinol Diabetes*. 2000;108(2):106-9. [\[PubMed\]](#)
- [9] Ziche M, Maglione D, Ribatti D, Morbidelli L, Lago CT, Battisti M, Paoletti I, Barra A, Tucci M, Parise G, Vincenti V, Granger HJ, Viglietto G, Persico MG. Placenta growth factor-1 is chemotactic, mitogenic, and angiogenic. *Lab Invest*. 1997 Apr;76(4):517-31. [\[PubMed\]](#)
- [10] Lei X, Zhang J, Shen J, Hu LM, Wu Y, Mou L, Xu G, Li W, Xu GT. EPO attenuates inflammatory cytokines by Muller cells in diabetic retinopathy. *Front Biosci (Elite Ed)*. 2011 Jan 01;3(1):201-11. [\[PubMed\]](#)
- [11] Vincent JA, Mohr S. Inhibition of caspase-1/interleukin-1beta signaling prevents degeneration of retinal capillaries in diabetes and galactosemia. *Diabetes*. 2007 Jan;56(1):224-30. [\[PubMed\]](#)