

*A project report on*

# **STUDENT LOGGER USING GAZE TRACKING**

*Submitted in Partial fulfillment for the award of the degree of*

**BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE  
ENGINEERING**

*By*

**RAM CHANDER(17BCE1038)**



**VIT<sup>®</sup>**  

---

**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

June, 2021



## **DECLARATION**

I hereby declare that the thesis entitled “**STUDENT LOGGER USING GAZE TRACKING**” submitted by me, for the award of the degree of Bachelor of Technology in Computer Science, VIT is a record of bonafide work carried out by me under the supervision of Satheesh Kumar.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place : Chennai

Date:

Signature of the Candidate

**Ram Chander**



## School of Computing Science and Engineering

### CERTIFICATE

This is to certify that the report entitled **STUDENT LOGGER USING GAZE TRACKING** is prepared and submitted by **Ram Chander(17BCE1038)** to VIT Chennai, in partial fulfillment of the requirement for the award of the degree of **B.Tech. CSE** programme is a bonafide record carried out under my guidance. The project fulfills the requirements as per the regulations of this University and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma and the same is certified.

#### Guide/Supervisor

Name: Sateesh Kumar B

Date:

Signature of the External Examiner

Name:

Date:

Signature of the Internal Examiner

Name:

Date:

Approved by the Head of Department, B. Tech CSE

Name: Dr. Justus S

Date:

(Seal of SCOPE)

## ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to Sateesh Kumar B, Associate Professor, SCOPE, Vellore Institute of Technology, for his constant guidance, continual encouragement, understanding; more than all, he taught me patience in my endeavor. My association with him is not confined to academics only, but it is a great opportunity on my part to work with an intellectual and expert in the field of Deep learning.

I would like to express my gratitude to Dr. G.Viswanathan, Mr. Sankar Viswanathan, Dr. Rambadu Kodali, Dr. Kanchana Bhaskaran V.S for providing an environment to work in and for his inspiration during the tenure of the course.

In a jubilant mood I express ingeniously my whole-hearted thanks to Dr. Justus S, Head of the Department & Co-ordinator, Dr. Karmel A, and Project Coordinator Dr. B V A N S S Prabhakar Rao, B. Tech. Computer Science and Engineering, SCOPE, VIT Chennai Campus, for their valuable support and encouragement to take up and complete the thesis.

Special mention to Dean, Dr. Jagadeesh Kannan R, Associate Dean, Dr. Geetha S, SCOPE, VIT Chennai, for spending their valuable time and efforts in sharing their knowledge and for helping us in every aspect.

All teaching staff and members working as limbs of our university for their not-self-centered enthusiasm coupled with timely encouragement showered on me with zeal, which prompted the acquisition of the requisite knowledge to finalize my course study successfully. I would like to thank my parents for their support.

It is indeed a pleasure to thank my friends who persuaded and encouraged me to take up and complete this task. At last but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project.

Place: Chennai

Date:

Name of the student

**Ram Chander**

## **ABSTRACT**

E-Learning is a booming industry and the COVID-19 pandemic only accelerated the rate at which online education became mainstream. The existing LMS platforms have covered most of the features like interaction, white board, attendance, etc. They missed out on one important feature though and that is the ability of the tutor to monitor and command the attention of a student in their class. This project will be useful for doing that as it constantly tracks the gaze of the user through a webcam to check whether the user is looking at the screen and paying attention to the class or is he looking away. We do this by creating a Neural Network architecture and passing values of the selected features into it.

Neural Network is a deep learning model and it has more layers like input layer, hidden layers and output layer. Each of these layers have neurons which are connected to neurons of other layers. We create a 4 layered Neural Network architecture to be able to find the result. This Neural network uses features like head tilt, head pose, face area, face coordinates, eye coordinates of the left and right eyes, centre of the pupil to find that out and this is done 6 times every second.

The project was designed to work with low level machines and give fast results with high accuracy and the project successfully manages to do that by having a high accuracy and also exhibiting live gaze tracking with minimum delay.

# **CONTENTS**

<b>DECLARATION</b>	<b>2</b>
<b>CERTIFICATE</b>	<b>3</b>
<b>ACKNOWLEDGEMENT</b>	<b>4</b>
<b>ABSTRACT</b>	<b>5</b>
<b>LIST OF FIGURES</b>	<b>8</b>
<b>LIST OF ACRONYMS</b>	<b>10</b>
<b>CHAPTER 1</b>	
<b>INTRODUCTION</b>	
1.1 INTRODUCTION	11
1.2 OBJECTIVES	12
1.3 BACKGROUND	13
1.4 MOTIVATION	15
<b>CHAPTER 2</b>	
<b>LITERATURE REVIEW</b>	
2.1 LITERATURE SURVEY	16
2.1.1 Gaze and Eye Tracking : Techniques and Application in ADAS	16
2.1.2 A Survey On Eye Gaze Tracking Techniques	17
2.1.3 NVGaze : Anatomically informed Dataset for Low Latency, Near Eye Gaze Estimation	17
2.1.4 Light-weight Head Pose Invariant Gaze Tracking	18
2.1.5 A Review And Analysis Of Eye Gaze Estimation systems, Algorithms and Performance Evaluation Methods in Consumer Platforms	19
2.1.6 Deep Multitask Gaze Estimation With A Landmark- Gaze Model	19

2.1.7 NVGaze: An Anatomically Informed Dataset	
For Low- Latency, Near Eye Gaze Estimation	20
2.2 EXPLORATION OF EXISTING MODELS	22
2.2.1 Introduction to Computer Vision	22
2.2.2 TensorFlow Object Detection Model	26
2.2.3 Head Pose Estimation	27
<b>CHAPTER 3</b>	
<b>DESIGN AND DETAILS</b>	
3.1 ENGINEERING DESIGN	29
3.2 SPECIFICATIONS	31
3.2.1 Software	31
3.2.2 Hardware	34
3.3 IMPLEMENTATION DETAILS	35
3.4 PROCEDURE	37
3.5 OUTPUT	39
<b>CHAPTER 4</b>	
<b>CONCLUSION</b>	
4.1 CHALLENGES AND CONSTRAINTS FACED	55
4.2 FUTURE IMPLEMENTATIONS	56
4.3 CONCLUSION	57
<b>REFERENCES</b>	<b>58</b>

## **LIST OF FIGURES**

1.1 Student Comparison on Quality of Teaching	12
2.1 ReLU Graph	22
2.2 Sigmoid Activation Function	24
2.3 Tanh Activation Function	25
2.4 Head Pose Estimation	27
3.1 Dataset	36
3.2 DLib Implementation	39
3.3 Check for Face in Grayscale	39
3.4 Face Detection	40
3.5 Eye points function	40
3.6 Masked Eye	41
3.7 Angle of the Head tilt	42
3.8 Function to get Bounding Box of face	42
3.9 Function with coordinates	43
3.10 Output with Bounding Box	43
3.11 3 Different Face Coordinates	44
3.12 Snippet for Head pose	44
3.13 Implementation of Head pose Algorithm	45
3.14 Output of Head Pose Algorithm	45
3.15 Output coordinates from algorithm	46
3.16 Snippet for Finding the most probable coordinates	46
3.17 Snippet using YOLO algorithm	47
3.18 Pupil Detection	47



3.19 Output with eye bounding box	48
3.20 Output as coordinates	48
3.21 Output after labelling and detection	49
3.22 Appending values into arrays	50
3.23 Appending values into arrays(2)	50
3.24 Arrays are added in new text file	51
3.25 Features are given as input	51
3.26 Result of the last 5 epochs	52
3.27 Snippet for Prediction of Output	53
3.28 Conversion to Tensors	53
3.29 Final Output	54

## **LIST OF ACRONYMS**

MOOC	Massive Open Online Courses
LMS	Learning Management System
ADAS	Advanced Driver Assistance System
CNN	Convolutional Neural Network
CLGM	Constrained Landmark Gaze Model
FOV	Field of View
OCR	Optical Character Recognition
CGI	Computer Generated Imagery
RGB	Red, Green and Blue
ReLU	Rectified Linear Unit
SSD	Single shot Multibox Detection
RAM	Random Access Memory
XML	Extensible Markup Language
MB	Megabytes
PnP	Perspective-n-Point
YOLO	You Only Look Once

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 INTRODUCTION**

This pandemic has challenged the education system to go far and beyond the current infrastructure in play. Unprecedented problems never dealt with before have presented themselves in a unique way, for both educators and the students. With virtual classrooms in play it has become increasingly difficult for teachers to monitor and assess the attention of students. This problem is not limited to classrooms as malpractice in online examinations continue to diminish the credibility of the assessment process.

This project will develop a mechanism to search through a dataset of images and classify them on the basis of their head tilt, head position, face area, face coordinates, eye coordinates of the left and right eyes, centre of the pupil and pupil position and track their gaze throughout the duration of surveillance.

Pictures are taken live from the webcam six times every second for continuous surveillance. The features of the head: face coordinates, facial area, head position and head tilt help in mapping the head which helps us figure out whether it is facing the screen or not and the features of the eyes such as the position of the eye, centre of the pupil, and eye coordinates help in figuring out where the gaze is directed towards. When these features are passed through a 4 Layer Neural Network architecture, we take the value from the sigmoid layer which determines whether the user is looking at the screen irrespective of the position of their head or eyes.

## **1.2 OBJECTIVES**

- The main aim is to continuously track the gaze of the user, and check whether he is looking at the screen precisely irrespective of the position of his head and eyes.
- To accomplish higher accuracy levels than existing solutions in an efficient manner.
- The method should be cost effective.
- Existing hardware of the user should be sufficient for using the application, hence there should be minimum hardware requirements for running this project.

## **1.3 BACKGROUND**

The COVID-19 pandemic resulted in all the educational institutions around the world getting shut down. The pandemic was supposed to end in a while and students were expected to come back to their respective schools and colleges for physical classes in the classroom. As the shutdown continued and it looked dangerous for students to attend classes, the institutions started conducting online classes. The online classes were an unexplored territory for most of the institutions and they were not in the position to handle it properly. They lacked the infrastructure required to have a good virtual classroom running on their own platform.

The education platforms created by technological giants like Google Classrooms and Microsoft Teams started being used widely across the globe as the platform for teaching students. The platform's user numbers went up astronomically, with Google Classroom now supporting over 150 million users and Microsoft Teams supporting over a 100 million users during the course of the year of 2020. These platforms while being very user friendly still lacked the feature of being able to notice whether the student was actually observing the class or not. Students are not designed to learn from the screen but

learn from the environment around them. They need real life observation of the tutors in order to be able to learn. While this is not possible through distance learning, what can be done is to simulate being observed through a gaze tracker like our project as it is found that eye contact with the tutor or learning materia is keen when it comes to learning for most of the children.

The rise of the global elearning market has been on the rise since prior to the pandemic. The market is expected to reach a whopping 336.98 billion U.S.D by the year of 2026, with a growth rate over 9% in the years 2018-2026. Another report has revealed that the number of students attending online classes for at least one of their undergraduate courses has gone from 15% in 2004 to 43% in 2016. The percentage of students who have done their entire undergraduate program online has also increased from 3.8% in 2008 to 10.8% in 2016. The reasons why a large number of people are gravitating towards online courses could be ranging from reluctance or inability to move from their locality, high tuition fee for actual in class college education, they are able to attend college and graduate from very reputed universities as the requirements for distance learning is much lesser compared to their in campus course and also the fact that almost all reputed universities have distance learning programs. Our project can be implemented for distance learning programs as well and will help the lecturers know who is attentive in class and who is not interested.

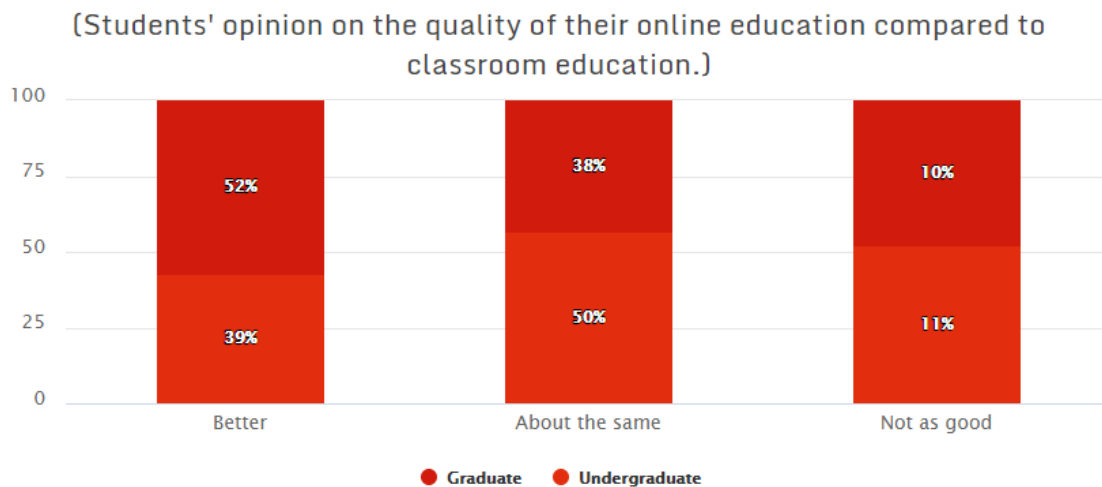
Other Online courses platforms and Massive Open Online Courses(MOOC) have also come up in recent times and they have a high number of students learning from them. There are also platforms for school children like Byjus, Unacademy, Panda, WhiteHat Jr which are conducting courses for primary to high school students and the parents are also enrolling them in such platforms. Online courses are provided by a lot of colleges in the US which students around the world are enrolling themselves in and get certificates for completing them at a very reasonable price. MOOCs like Coursera and Udacity have also recorded the highest number of users over the last 3 years and are adding more courses in various different fields as well, diversifying their user base. Our software can be applicable to any of these platforms as well and it will be useful especially in the case of

younger kids and school students who learn from here as their parents can actually check for how long the kid was paying attention to the class.

Checking for malpracticing in online examination is another big area that none of the already existing platforms have as a functionality. With our project, the institution can conduct quizzes and multiple choice questions, which is already the mode in which most of the universities and MOOCs conduct their online assessments , and find out who is looking away from the screen and for what duration and take necessary action if they find it to be malpractice. The project has a huge scope and the further developments may also allow the students to take notes or write in their notebooks and not consider that to be a malpractice, it is anyway what the institution requires as this project is designed to be highly flexible and adaptable.

## 1.4 MOTIVATION

A scenario where all the educational institutions and organisations get closed down, has never happened before so naturally there were a lot of loopholes in the new infrastructure that everyone tried to create for taking classes. This provided an opportunity for me to go ahead and fix one of the loopholes that I feel will help in the creation of a better learning experience for all the students and children who will end up using this system. The survey also shows that the E-learning system is going to continue growing and more people are gravitating towards it. Our software will be a good tool for all these organisations and parents to have in order to monitor the students and kids.



*Fig 1.1: Students Comparison in Quality*

This system will also ensure that malpractices do not occur during the assessments as that seems to be a problem throughout all platforms. Since physical invigilation is not possible in distance learning a lot of malpractices happen through looking at the phone or some other screen/ material since most of the platforms and institutions have managed to block moving out to another window during an examination. This system may also help in eliminating such fraudsters and establish an automated method for exam invigilation.

## **CHAPTER 2**

### **LITERATURE REVIEW**

#### **2.1 LITERATURE SURVEY**

##### **1. Gaze and Eye Tracking : Techniques and Application in ADAS [1]**

Authors : Muhammad Qasim Khan

Sukhan Lee

Tracking drivers' eyes and gaze could be a topic of good interest in advanced driver help systems (ADAS) analysis. especially, it's seriously mentioned within the community of traffic safety researchers, since visual distraction is taken into account together with the most causes of traffic accidents. During this paper, they have a tendency to initially comprehensively review eye-tracking and gaze-tracking techniques and discuss their main classes. the benefits and limitations of every class area unit explained in terms of their needs and sensible applications. In another section of the paper, the applications of eye and gaze chase systems in ADAS area units are mentioned. method of aggregation driver eye and gaze information and also the algorithms wont to process this information area unit explained. It'll be explained however the information associated with the driver's eyes and gaze is utilized in ADAS to scale back the losses related to traffic accidents caused by driver's visual distraction. A discussion of the specified options of current and future eye and gaze trackers is additionally bestowed.



2. A Survey On Eye Gaze Tracking Techniques [2]

Authors: H.R.Chennamma

Xiaohui Yuan

This paper reviews the latest techniques in the field of gaze tracking as it is one of the most challenging tasks of Computer Vision. With the introduction of different strategies of eye-tracking, they've awarded a review of non-contact video-based mostly eye-tracking. The intention of this paper is to manage a review of recent growth in non-contact video-based eye-tracking. Although eye-tracking encompasses a history of a hundred years of analysis, it has not been standardized. Future developments in eye-tracking need to focus on standardizing which eye movement metrics are used, how they are named, and how they need to be understood in the context of interface style (Poole et al. (2004)). For example, there is not yet a standard for the minimum duration of a fixation. The intrusiveness of the kit should be shrunk to make users feel easier. The strength and accuracy of information capture needs to be improved and eye-tracking systems need to become cheaper to make them a viable usability tool for smaller industry agencies and analysis laboratories.

3. NVGaze : Anatomically informed Dataset for Low Latency, Near Eye Gaze Estimation [3]

Authors: Joohwan Kim

Michael Stengel

Alexander Majercik

Shalini De Mello

David Dunn

Samuli Laine

Morgan McGuire

David Luebke

Quality, diversity, and size of the coaching dataset are important factors for

learning-based gaze estimators. They tend to generate two datasets that meet these criteria for near-eye gaze estimation under infrared illumination: an artificial dataset using anatomically informed eye and face models with variations in face shape, gaze direction, pupil and iris, skin tone, and external conditions (two million images at 1280x960), and a real dataset collected with thirty-five subjects (2.5 million images at 640x480). Exploiting their datasets, they tend to train a neural network for gaze estimation, achieving a pair of 0.06 ( 0.44) degree accuracy over a good thirty x forty degree reading field on real subjects excluded from training, and 0.5 degree best-case accuracy (over a similar field of view) as soon as explicitly trained for a real subject. They have a tendency to train jointly a variant of our network to perform pupil estimation that exhibits higher resilience than previous ways. The network requires fewer convolutional layers than previous networks and achieves sub-millisecond latency.

#### 4. Light-weight Head Pose Invariant Gaze Tracking [4]

Authors: Rajeev Ranjan

Shalini De Mello

Jan Kautz

Unrestricted remote gaze tracking exploiting ready cameras could be a difficult drawback. Recently, promising algorithms for appearance-based gaze estimation exploitation convolutional neural networks (CNN) are projected. increase their resilience to numerous conflicting factors together with variable head cause, subject identity, illumination and image quality stay open questions. During this work, they examine the result of variable head cause trained on machine learning regressors to estimate gaze direction. They tend to propose a unique branched CNN design that improves the robustness of gaze classifiers to variable head cause while not increasing process price. They tend to additionally give varied procedures to effectively train our gaze network along with transfer learning from the additional closely related task of object viewpoint estimation and from an oversized hi-fi artificial gaze data set that changes our 10 times faster gaze

network to achieve competitive accuracy to its current state-of-the-art direct competitor.

5. A Review And Analysis Of Eye Gaze Estimation systems, Algorithms and Performance Evaluation Methods in Consumer Platforms [5]

Author : Anuradha Kar

In this paper, she reviews the analysis of techniques and applications for gaze direction determination, which has evolved in various ways over the last 20 years. Several generic use cases for gaze tracking are identified: Desktop, TV, Headmounted, Automotive and Handheld devices. The analysis of the literature ends in the identification of many platform-specific factors that affect the accuracy of gaze tracking. A major outcome of this review is the realization of the desire to develop standardized methods for performance analysis of gaze tracking systems and to achieve consistency in their specification and comparative analysis. To fulfill this desire, the construction of a methodological framework for meaningful analysis of different gaze tracking systems is planned.

6. Deep Multitask Gaze Estimation With A Landmark- Gaze Model [6]

Authors: Yuyu

Gang Liu

Jean-Marc Odobez

As an indicator of attention, gaze is a very important cue for analyzing human behavior and social interaction. Recent deep learning methods for gaze estimation rely on a simple regression of gaze from images, and do not take into account the potential inconsistencies in cropping and standardizing eye images. This can impact the estimation of implicit relationship between visual cues and also the gaze direction once managing low resolution images or once coaching with a

limited amount of information. During this paper, they have planned a tendency to propose a deep multitask framework for gaze estimation, with the following contributions. i) they have planned a tendency to propose a multitask framework that depends on each artificial information and real information for end-to-end coaching. During training, each dataset provides the label of just 1 task, but the 2 tasks are combined in an extremely strained means. ii) They have a tendency to introduce a constrained Landmark-Gaze

Model (CLGM) that models joint variation in the positions of eye landmarks (including the iris center) and gaze directions. By linking explicitly visual data (landmarks) to the additional abstract gaze values, they have a tendency to show that the computer is additionally correct and easier to figure out. iii) By turning our deep network into one that jointly derives the CLGM model parameters and the eye region scaling and translation parameters on the one hand, and a CLGM-based mostly decoder that deterministically derives the landmark positions and gaze from these parameters and head generation on the other hand, our framework decouples gaze estimation from irrelevant geometric variations in the eye image (scaling, translation), resulting in an additional strong model. Thorough experiments on public datasets show that our technique achieves competitive results, over progressive results in difficult free head create gaze estimation tasks and on eye landmark localization (iris location) ones.

## 7. NVGaze: An Anatomically Informed Dataset For Low- Latency, Near Eye Gaze Estimation [7]

Authors: Joohwan Kim

Shalini De Mello

Michael Stengel

Alexander Majercik

David Dunn

Samuli Laine

Morgan McGuire

David Luebke

Quality, diversity, and size of coaching knowledge are critical factors for learning-based gaze estimators. We tend to generate 2 datasets that meet these criteria for near range gaze estimation under infrared illumination: an systemic data using anatomically trained eye and face models with variations in face shape, gaze direction, pupil and iris, skin tone, and external condition (2 million images at 1280x960), and a real dataset collected with thirty five subjects (2.5 million images at 640x480). We use these datasets to train neural networks that perform with latency less than one millisecond. Our gaze estimation network achieves 2.06( $\pm 0.44$ ) of accuracy over a wide 30x40 degree field of view on real subjects excluded from training and 0.5% of best case accuracy (a constant FOV) explicitly trained on real subjects. We also train a pupil localization network that achieves higher accuracy than previous strategies.

## 2.2 EXPLORATION OF EXISTING MODELS

### 1. INTRODUCTION TO COMPUTER VISION [8]

Computer vision is a rapidly evolving field which ended up becoming one the primary use cases of Neural Networks. Computer Vision uses Deep Learning primarily and the functional possibilities for Computer Vision has also increased at a great pace over the last 10 years. Some of the uses are:

1. Optical character recognition (OCR)
2. Automation in billing
3. Medical field
4. Safety in Automobiles(Gaze tracking)
5. Entertainment Industry (e.g. merging CGI with live actors in movies)
6. Motion capture (mocap)
7. Surveillance
8. Biometrics

Convolutional Neural Networks are used to achieve these tasks. The necessary components for building such CNNs are given below.

1. Convolutional kernel
2. Striding
3. Padding
4. Pooling: There are two types of Pooling

(i)

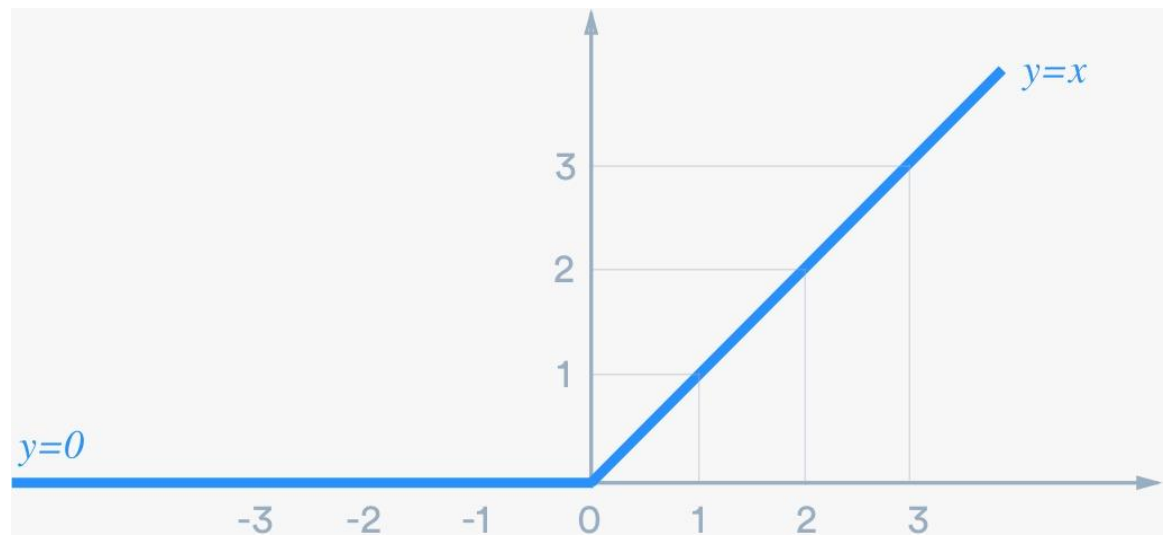
(i)Max Pooling

(ii) Average Pooling

## 5. RGB input

### ACTIVATION FUNCTIONS

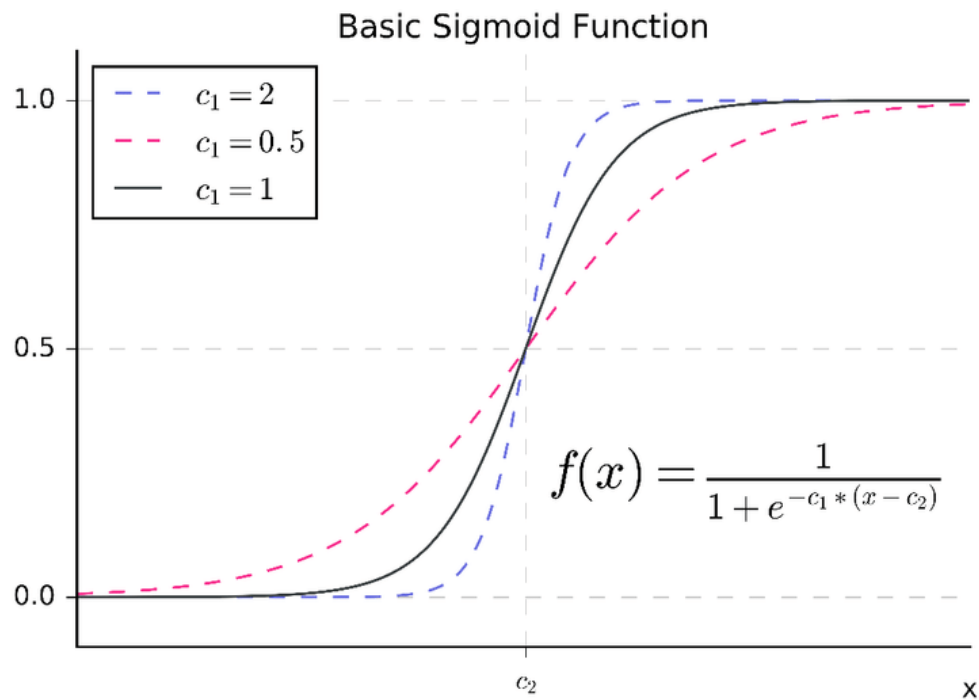
Rectified Linear Unit (ReLU): The ReLU function takes any negative value and declares it as zero whereas any positive value continues to be considered by the same value.



*Fig 2.1 : ReLU Graph*

### Sigmoid Function

Sigmoid functions are the output layer functions of the model. They are non linear in nature and are non linear in nature. They are predominantly used in feedforward neural networking models. They are represented in graph format as the following :

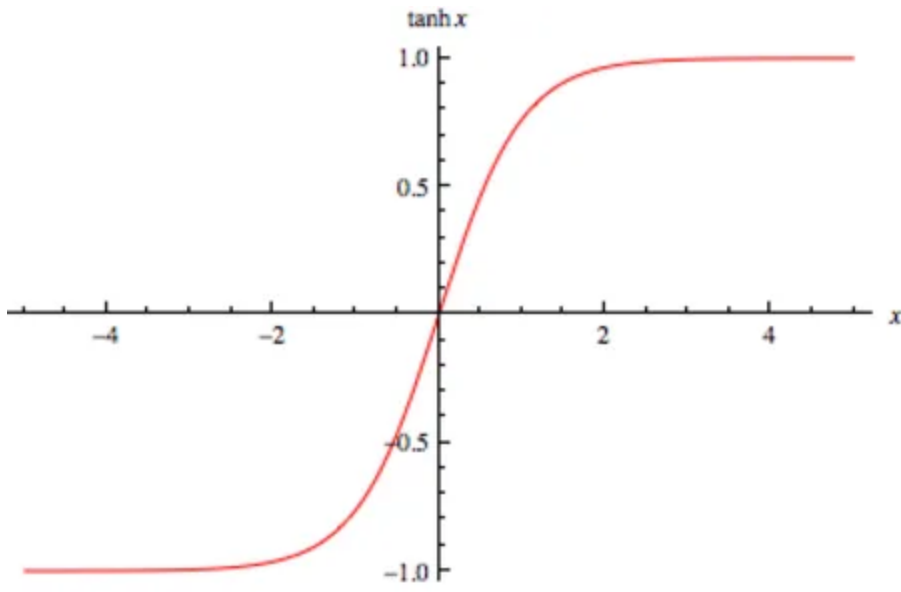


*Fig 2.2 : Basic Sigmoid Function*

### Hyperbolic Tangent Function(tanh)

This function is an alternative for the sigmoid function and has a range from -1 to 1. They are far smoother in use and take in any real value as a valid input. They have a similar S Shaped function as well. Tanh is just considered to be an upscaled sigmoid function.





*fig 2.3 Tanh*

## 2. TENSORFLOW OBJECT DETECTION MODEL [9]

We will be checking a few already existing TensorFlow Object detection models to see which one will be more applicable to us for using in this project. We explore the following models with our dataset and come to certain conclusions :

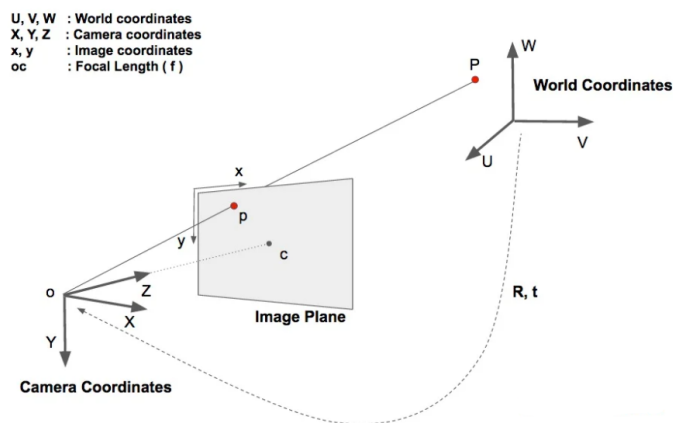
- a. MobileNet and SSD integration : MobileNet in SSD framework is highly efficient as they are very easy to run deep neural networks even on low end devices like mobile phones and laptops and work beautifully for real-time applications such as our project. The Mobilenet has separated channels in their convolutional layers, this results in them being much faster than the standard convolutional layers used as they have fewer multiplication and addition operations. Since MobileNet is quick and boasts a good accuracy rate as well.
- b. Inception : Inception is a heavily engineered network that provided ground breaking results in terms of both speed and accuracy. While most convolution layers were stacked deeper and deeper by CNNs, Inception brought in the concept of classifiers and increased the performance drastically. Through running the dataset on this model, we realise that our accuracy was significantly higher than MobileNet but the speed was very low as it has a far more deeper neural network.

Conclusion : We decided to use the MobileNet SSD model because the project needs to work for real time applications and hence the speed is also very important. The Inception despite having higher accuracy is not as fast as MobileNet SSD and is not as friendly as MobileNet SSD is to low end devices which is what most of the students will be using to attend classes as well.

### 3. HEAD POSE ESTIMATION

The pose estimation is referred to as Perspective-n-Point in computer vision language. The locations of 3D points of the project are known to us by the 2D projections of the object. A 3D object only has two kinds of motions with respect to a camera: one is rotational and the other is translation. You can calculate the 3D pose of an object using the coordinates of a few 2D points and the coordinates of the same 3D points. For a face, Using the tip of the nose, corners of the mouth, the tip of the chin and the corners of the eye gives best results. In order to get the 3D coordinates for the same point, we can just take one of the points and keep them as the reference point. For a face, taking the tip of the nose as your reference point is a good trick.

The pose estimation algorithm uses the 3D coordinates that we have obtained in the world plane and using the camera's properties like depth, focal length and optical center, it projects it onto the image plane.



*Fig 2.4 Head Pose Estimation*

In this figure, O is the center of the camera and it exists on the image plane. Now, we have to find out the formula for projection of point P onto the image plane. This can be found out as we know the location of point P (u,v,w) in world coordinates and using the 3x3 Rotational matrix and 3x1 translation matrix of the world coordinates we can find out (x,y,z) of the point P in camera coordinate system using the formula given below :

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \mathbf{R} \begin{bmatrix} U \\ V \\ W \end{bmatrix} + \mathbf{t}$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = [\mathbf{R} \mid \mathbf{t}] \begin{bmatrix} U \\ V \\ W \\ 1 \end{bmatrix}$$

Since we know (X,Y,Z) and (U,V,W), we can find out  $r_{ij}$  and  $(t_x, t_y, t_z)$  .

## **CHAPTER 3**

### **DESIGN AND DETAILS**

#### **3.1 ENGINEERING DESIGN**

The basic steps taken into account are

1. **Project Definition :** The aim of the project was gaze tracking to check whether the user is looking at the screen or not, by using the webcam camera that gets 6 frames per second to have a video like surveillance.
2. **Background Research :** The Student Logger for Gaze tracker has never been done before and such a system is a necessity for the market now as well. Other gaze tracking models are either very slow and hence cannot check for 6 frames per second or they require a lot of hardware and are costly in which case they cannot be used as a student logger system because we want to run this project with minimum hardware and software requirements.
3. **Gap Analysis :** The COVID-19 pandemic was not expected and the educational institutions all around the world did not have the necessary infrastructure to run online classes and examinations. While users for LMS platforms have soared in numbers there is a feature that none of them offer and that is surveillance. In a class, the students are under the observation of the lecturer but that is not the case with online classes, hence there is a need in the market for this software.
4. **Proposed Solution :** Using the images taken from the webcam, we can find out the face coordinates, head pose, tilt of the head and the pupil coordinates which can be passed as features into a 4 layer neural network. In which we can take values from the sigmoid layer and use that to find out.

5. Solution and Validation : The value from the sigmoid layer will be between 0 and 1, if the value is lesser than 0.5 then we can pass it as not seeing the screen and otherwise he is looking at the screen. The output is shown as 'Yes' or 'No' on the screen of the user itself and can be stored or passed and used in any way necessary for future applications as well.

### 3.2 SPECIFICATIONS

There are two types of Specifications which are as follows:

#### 1. SOFTWARE SPECIFICATIONS :

- a. Python : It is a high level programming language that can be used in any system with the aim of helping programmers create high level object oriented programs with better clarity in logic for small and big scale projects. The language has very bare minimum requirements such as 90 Mb of RAM and very little storage space like 25 Mb, that is the reason they use this language to code on Raspberry Pi, etc. It is a language that is easily understandable and comes with a lot of modules that we can use in our project. The modules that we will be using in our project are:
  - i. Pandas : Pandas may be a powerful and versatile Python package that enables you to figure with tagged and statistical information. It additionally provides statistics strategies, allows plotting, and more. One important feature that we use in our project is its ability to read and write plenty of different sorts of files.
  - ii. Numpy : Numpy gives us a lot of high level mathematical functions that we can use on our data and a lot of array manipulations that can be done using this module. We use Numpy to store and operate on data by the computer vision module named OpenCV. Since we have to find the coordinates of the bounding boxes that we apply to the face and eyes and the other values that we retrieve, we will be using Numpy for all of those purposes.
  - iii. OpenCV : OpenCV is primarily a library consisting of functions that can be used for image manipulation. It is often used in Computer Vision based projects This module has been used to help

us find the coordinates of the boxes and then the position of the eye, the angle of the head and these values are what we use in our Neural Network Architecture to obtain an output. This handles all of our image manipulation apart from face landmark detection.

- iv. Dlib : Dlib is a module that we use for facial landmark detection in our project. It is very light weight and it is the fastest method for face detection. This module can work with even lower resolution and that is why we use this to process the images at a lower resolution in order to find the Left and Right Eye socket.
- v. Imutils : Imutils are functions that are there to make it more convenient for us to do image processing. With this module we will be able to resize, rotate, skeletize and draw Matplotlib images. We use this in our project to draw bounding boxes on the faces after we find the sockets and their coordinates.
- vi. LabelImg : LabelImg is a graphical image annotation tool. It's written in Python and uses Qt for its graphical interface. Annotations are saved as XML files in PASCAL VOC format, the format employed by ImageNet. We use this to mark annotations in Object Detection.
- vii. TensorFlow : TensorFlow is an open source library that has a large collection of algorithms and models that help with projects related to machine learning and deep learning. We can do large scale machine learning processes and high numeric computations using this library. This library was developed by Google's Brain Team. It has a large set of uses ranging from Image recognition, NLP, PDE based simulations and Neural Networking. We use this in our



project for two purposes :

1. Artificial Neural Networks : Since Neural Network has nodes and our neural network architecture has 4 layers we use TensorFlow to make it more convenient. Since Tensorflow uses python for providing all of it's features, all the nodes and tensors become objects of Python itself and hence TensorFlow Applications become Python applications as well. This results in ease of using data of objects and nodes across the program.
  2. Object Detection : We use the TensorFlow Object Detection API for detecting the pupil in our images.
- b. Operating System : Python can be run on any modern operating system but since there is a lot of processing that is being done we would suggest any basic laptop's operating system such as :
- i. Microsoft XP and later
  - ii. MAC OS X 10.11 or higher, 64 bit
  - iii. Linux RHEL 6/7, 64 bit (All libraries work on Ubuntu as well)

## 2. HARDWARE :

- a. The Processor: x86 64 bit CPU (Intel/ AMD Structure) works the best for this
- b. Random Access Memory(RAM) : 2 GB will be sufficient for executing this efficiently but if the RAM is higher the speed will be better.
- c. Webcam : A basic laptop webcam will be necessary to run this project because otherwise we will not get the input to check from anywhere. Any laptop's webcam will work with this project, that is why this project is universally applicable. The image is usually 480x640 which is what we use in the project.

### 3.3 IMPLEMENTATION DETAILS

The dataset that we are using for this project is self made. The data set has a total of 30000 images of people. The dataset is classified into two types, the first type is the one where the user is looking at the screen and the second type is the one where the gaze of the user is not on the screen. We have collected over 15000 images for each of the 2 classes and trained them using a neural network model.

We trained the data set by telling the system that we will be looking at the screen for the next  $n$  images, once the program starts running, the webcam continuously tracks the user by taking 6 images every second and computes all the features and feeds it into the neural network to be considered as looking at the screen. The vice versa is done when you are training the model to spot when the user is not looking at the screen.

The storage for this program is not much because even though we are working with images and storing the data of the image, the image itself is not stored. We have a file for the dataset where only the necessary values are stored with an ending of either 1 or 0 signifying where they were watching the screen or not respectively.

273	342	283	353	239	343	250	356	288	335	298	347	-8.84	203	300	125	124	264	385	173	404	1
287	334	298	347	236	346	248	356	276	339	287	352	-8.65	203	300	125	124	264	382	172	401	1
287	330	298	340	281	331	293	343	237	332	248	344	-8.47	203	300	125	124	264	373	-1427	437	1
286	321	297	332	274	324	285	337	243	326	255	340	-9.87	203	286	125	125	263	360	393	475	1
275	319	284	329	239	324	250	335	237	323	249	333	-9.87	203	286	125	125	262	348	380	476	1
287	308	297	319	240	319	252	329	275	314	285	325	-11.56	203	272	125	125	262	337	369	481	1
268	308	277	317	244	307	255	318	241	311	253	320	-10.54	227	273	103	104	264	327	390	478	1
286	302	297	314	290	300	303	311	244	304	256	317	-10.3	227	273	103	104	265	323	374	478	1
278	307	288	316	245	307	256	317	243	317	255	329	-10.3	227	273	103	104	266	322	436	465	1
246	316	258	329	248	305	258	315	246	307	258	316	-6.34	227	273	103	104	273	319	353	485	1
261	303	272	312	295	305	304	314	258	304	269	313	0.0	238	273	104	104	287	318	254	280	1
278	299	290	310	312	304	322	314	280	301	293	312	7.77	261	273	104	104	309	320	363	290	1
288	299	300	307	339	306	348	315	289	301	299	310	12.8	261	284	104	104	321	322	271	484	1
333	307	343	316	342	307	352	318	295	299	306	311	14.04	273	273	104	104	324	323	407	302	1
295	300	306	311	328	307	338	316	330	310	341	318	14.04	273	284	104	104	327	324	414	308	1
388	305	401	321	384	308	394	323	376	309	386	325	-7.43	314	272	124	125	388	345	314	488	0
387	307	399	319	384	311	394	321	344	312	355	325	-6.79	314	272	124	125	388	344	256	474	0
382	305	394	320	378	308	389	321	333	307	344	322	-7.13	300	272	124	125	383	339	310	487	0
332	309	344	321	329	312	341	321	386	305	397	315	-8.13	300	272	124	125	384	338	512	320	0
331	308	343	322	386	304	396	316	330	313	343	325	-8.3	300	272	124	125	384	339	300	485	0
381	305	393	320	332	306	344	322	331	313	344	326	-7.28	300	272	124	125	382	338	344	485	0
331	305	343	319	382	299	394	315	329	309	341	320	-8.3	300	272	124	125	384	336	378	482	0
328	305	340	317	388	301	400	314	380	304	391	315	-7.39	300	272	124	125	381	335	248	469	0
329	301	340	315	379	297	391	311	376	298	388	312	-6.58	300	272	124	125	378	331	362	485	0
375	300	388	313	327	301	338	314	369	302	379	314	-3.58	300	272	124	125	377	329	475	303	0
327	301	338	314	375	299	388	312	367	301	378	313	-4.76	300	272	124	125	376	328	277	485	0
379	298	389	307	331	305	343	318	196	357	277	377	8.47	300	259	124	124	373	329	292	485	0
332	304	343	318	193	357	275	377	332	295	344	304	8.47	300	259	124	124	372	327	543	300	0
375	298	384	306	327	303	339	315	196	359	275	378	9.66	300	259	124	124	371	326	1259	1898	0
377	294	388	303	368	293	378	303	327	290	338	300	5.83	300	259	124	124	369	326	547	305	0
379	299	390	307	379	300	391	310	364	297	373	305	9.64	300	259	124	124	369	326	520	298	0
379	295	390	304	326	287	337	295	195	358	275	377	8.3	300	259	124	124	366	323	358	342	0
375	311	384	319	376	309	387	319	375	312	386	322	5.71	300	272	124	125	357	330	361	286	0
193	359	273	378	621	433	636	449	316	326	327	339	3.9	296	284	104	104	358	331	447	294	0
190	359	272	380	362	331	374	343	323	330	335	343	0.0	296	284	104	104	352	330	428	299	0
189	360	271	380	314	333	326	346	362	332	374	344	2.94	296	284	104	104	348	331	384	294	0

Fig 3.1 Dataset

### 3.4 PROCEDURE

#### OVERVIEW :

1. Image from the Webcam is retrieved :
2. Face related data retrieval :
  - a. The image is passed into Dlib for predicting the landmark points on the face.
  - b. Left and right eye regions are retrieved from the landmark detections.
  - c. The line joining the centers of the 2 eyes is compared with a horizontal line to find the angle of tilt of head.
  - d. The eye region is then masked out from the image for faster implementation of pupil detection (object detection task)
  - e. A bounding box is drawn surrounding the face to calculate the area of the face region
  - f. Head pose estimation is computed to find the line of alignment of head
3. Pupil Tracking : The masked eye image gets passed into the MobileNet SSD model, we use this model to generate the coordinates of the pupil. This MobileNet model is chosen because it is faster and accurate and

when compared to other Object detection models. It's speed and ability to work with low end devices make it the perfect choice for our project where we have to track live video and can't have heavy requirements either.

- a. Using MobileNet model find the bounding box of the pupil
- b. Use the SSD object detection algorithm and find the Pupil center.

4. The final gaze prediction:

- a. Pass the following retrieved values into a 4 layered Neural Network :
  - i. Face bounding box coordinates
  - ii. Face Area
  - iii. Pupil bounding box coordinates
  - iv. Pose of head
  - v. Head tilt angle
- b. The output layer with a sigmoid activation function returns a value between 0 and 1:
  - i. If the value is below 0.5 - Not seeing the screen
  - ii. If the value is above 0.5 - Seeing the screen

### 3.5 OUTPUT

First the dlib facial landmark is initialized. Then it's model weights which predict the 68 landmark points(including eye region) on the face are loaded. Second, the ssd object detection model for pupil detection is loaded. A config file for pupil detection after its paths are initialised has the model architecture. Similarly, the checkpoint file has it's weights. Then we load our final gaze prediction model that is a 4 layer neural network along with it's weights.

```
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor('track_dependencies\shape_predictor_68_face_landmarks.dat')
CONFIG_PATH = 'pupil track dependencies/pipeline.config'
CHECKPOINT_PATH = 'pupil track dependencies/'
WEIGHT_PATH = "4layer/last2.hdf5"

configs = config_util.get_configs_from_pipeline_file(CONFIG_PATH)
detection_model = model_builder.build(model_config=configs['model'], is_training=False)
ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(CHECKPOINT_PATH, 'ckpt-7')).expect_partial()
category_index = label_map_util.create_category_index_from_labelmap('pupil track dependencies/label_map.pbtxt')
model = load_model('4layer/Model.h5')
model.load_weights(WEIGHT_PATH)
```

*Fig 3.2: dlib*

Webcam image is captured and converted to grayscale for faster performance. Then the initialized Dlib model checks for faces in the grayscale image.

```
while True:
    _, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = detector(gray)
    key = cv2.waitKey(1)
```

*Fig 3.3 : Dlib Model checks for Face in grayscale*

If the number of detected faces is greater than or equal to 1, the landmark points on the face are predicted. Then the predicted landmarks are passed along with the image in a new function that finds the tilt of the head and prints a cropped image of the eye region. If the number of detected faces is less than 1, the application waits until a face is detected.

```
while True:
    _, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = detector(gray)
    key = cv2.waitKey(1)

    if (len(faces) >= 1):
        for face in faces:
            landmarks = predictor(gray, face)
            CC, angle = get_eye_points(frame, landmarks)
            print(angle)
```

*Fig 3.4: Face Detection*

In this function `get_eye_points`, with the parameters as image and facial landmarks is called, the landmark points of the left eye and right eye are isolated. The left eye points are considered to be between 36 to 41 and the right eye from 42 to 47. These coordinates are painted in white over an empty image to create a mask. A bitwise and operator on the input image and the mask is applied to isolate the eye region. The masked eye image is then passed into an object detection algorithm for pupil detection.

Also, the isolated left and right eye points can be used to get the center of the 2 eyes. Therefore, a new line that connects the centers of the 2 eyes can be compared with a horizontal line, the angle between these 2 lines is the angle of tilt of the head. Also the masked eye image from this function is passed in the object detection algorithm that outputs the coordinates of the bounding box of a pupil.



Code snippet of get\_eye\_points function:

```
def get_eye_points(img, facial_landmarks):
    lp = [36, 37, 38, 39, 40, 41]
    rp = [42, 43, 44, 45, 46, 47]
    mask = np.zeros((480, 640), np.uint8)

    lr = np.array([(facial_landmarks.part(point).x, facial_landmarks.part(point).y) for point in lp])
    rr = np.array([(facial_landmarks.part(point).x, facial_landmarks.part(point).y) for point in rp])

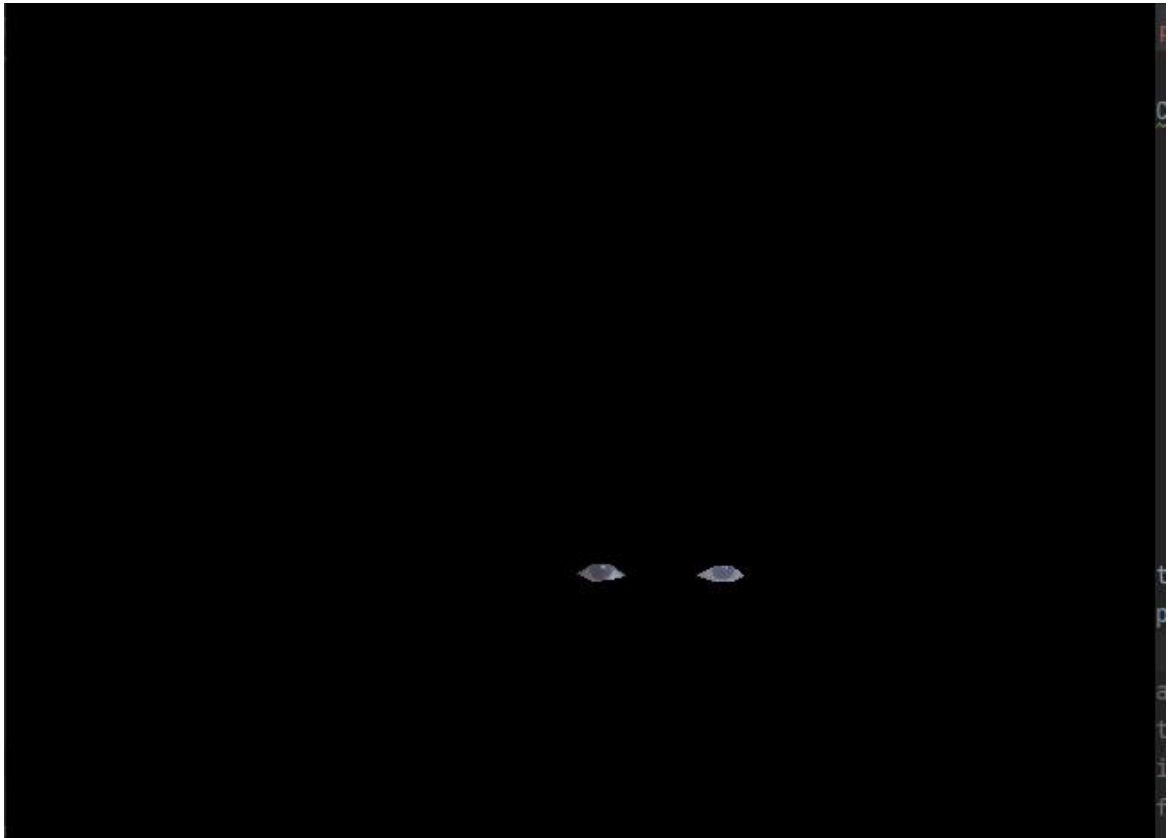
    mask1 = cv2.fillPoly(mask, [lr], 255)
    mask1 = cv2.fillPoly(mask1, [rr], 255)
    eye = cv2.bitwise_and(img, img, mask=mask1)

    leftEyeCenter = lr.mean(axis=0).astype("int")
    rightEyeCenter = rr.mean(axis=0).astype("int")
    dY = rightEyeCenter[1] - leftEyeCenter[1]
    dX = rightEyeCenter[0] - leftEyeCenter[0]
    angle = np.degrees(np.arctan2(dY, dX))
    angle = round(angle, 2)

    return eye, angle
```

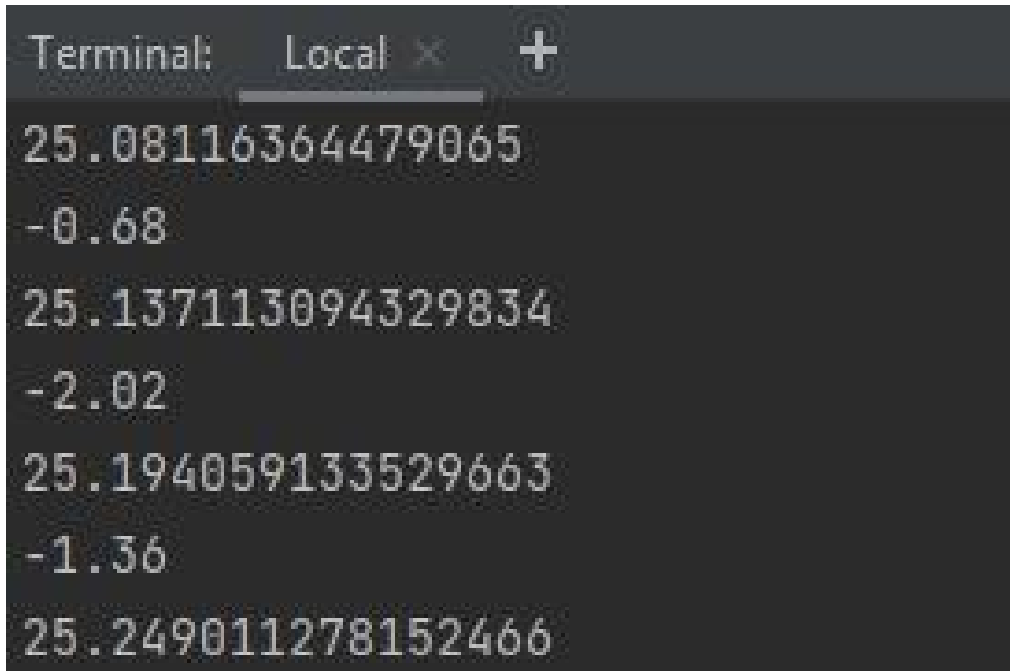
*Fig 3.5: Eye points Function*

The masked eye image:



*Fig 3.6: The Masked Eye Image*

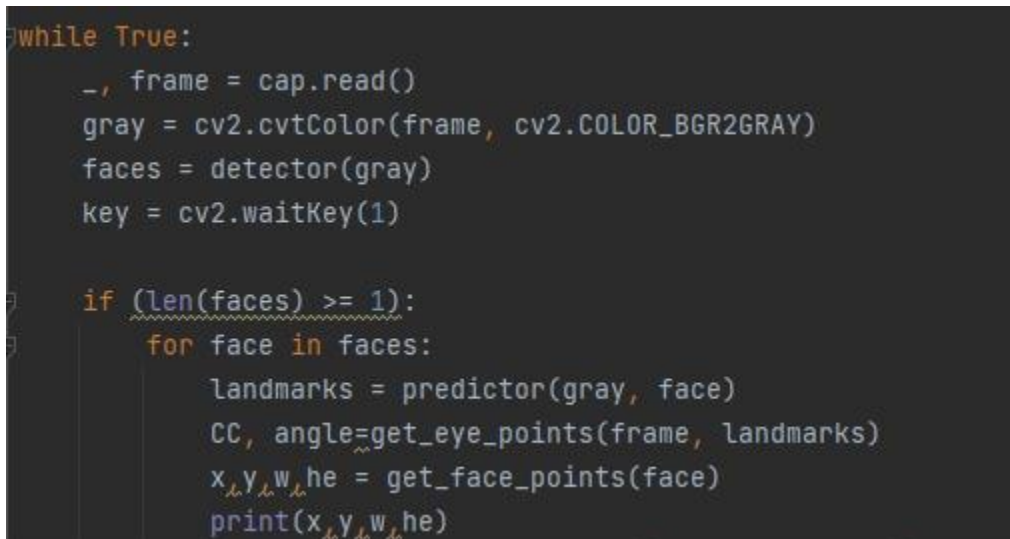
Angle of tilt of head for each timestamp :



```
Terminal: Local x +
25.08116364479065
-0.68
25.137113094329834
-2.02
25.194059133529663
-1.36
25.249011278152466
```

*Fig 3.7: Angle of Head tilt*

The detected face from dlib is used to get a bounding box on the face using the function `get_face_points(face)`



```
while True:
    _, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = detector(gray)
    key = cv2.waitKey(1)

    if (len(faces) >= 1):
        for face in faces:
            landmarks = predictor(gray, face)
            CC, angle = get_eye_points(frame, landmarks)
            x, y, w, h = get_face_points(face)
            print(x, y, w, h)
```

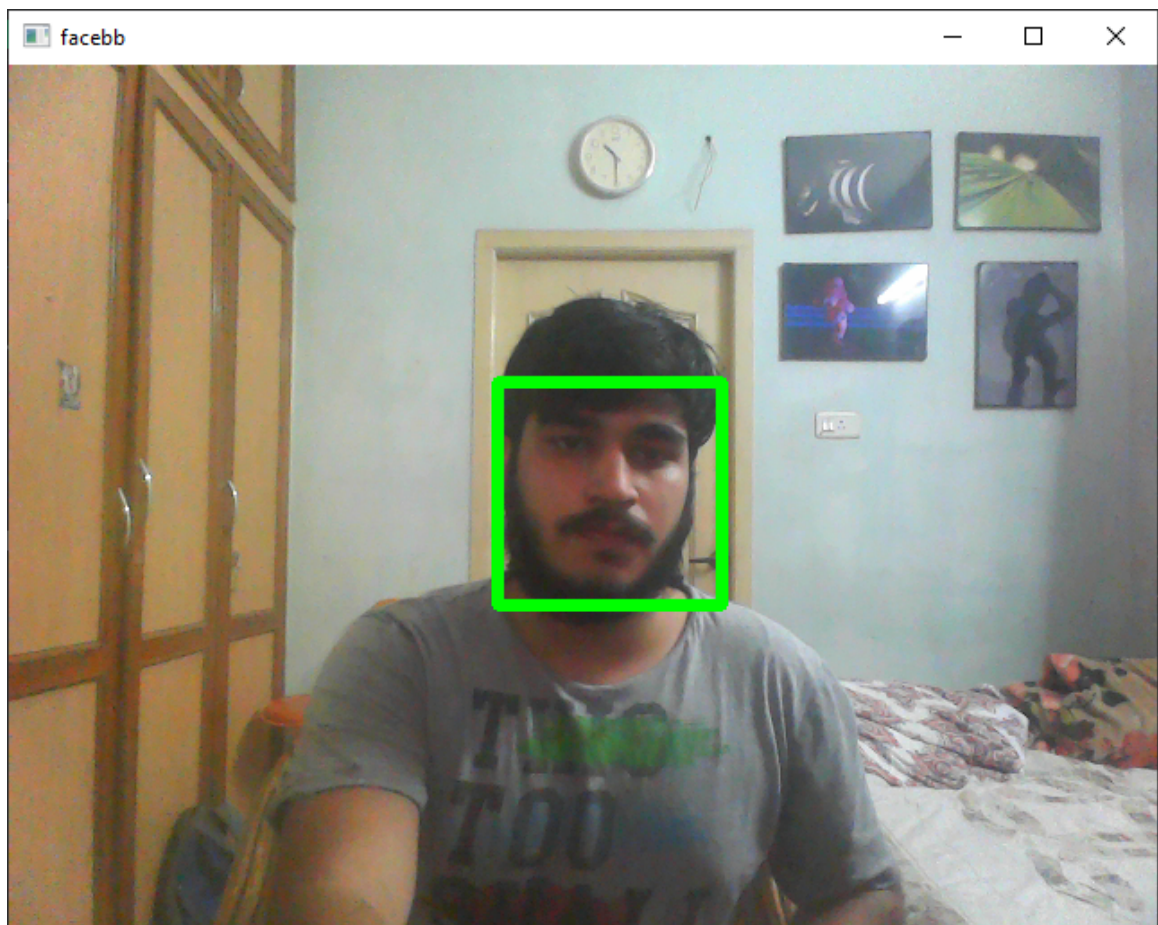
*Fig 3.8: Function to get Bounding box on face*

The `face_utils` is a built-in function from the `imutils` library that converts the landmark position of `dlib` into a bounding box. To do this the maximum and minimum of `x` and `y` coordinates of the landmarks are retrieved. With this a rectangle is drawn.

```
def get_face_points(face):  
    (x, y, w, h) = face_utils.rect_to_bb(face)  
    return x, y, w, h
```

*Fig 3.9: Function to get Coordinates*

The outputs of the above function:



*Fig 3.10: Output with Bounding Box*

```
37.1726598739624
311 294 149 149
37.22459435462952
311 294 149 149
37.28393840789795
311 294 149 149
```

*Fig 3.11: 3 Different face coordinates*

Next the head pose or the line at which the head is aligned is retrieved

```
while True:
    _, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = detector(gray)
    key = cv2.waitKey(1)

    if (len(faces) >= 1):
        for face in faces:
            landmarks = predictor(gray, face)
            CC, angle = get_eye_points(frame, landmarks)
            x, y, w, h = get_face_points(face)
            x0, y0, x1, y1 = get_head_pose(gray, landmarks)
            print(x0, y0, x1, y1)
            image = cv2.line(frame, (x0, y0), (x1, y1), (200, 40, 0), 2)
            cv2.imshow('head pose', image)
```

*Fig 3.12 : Snippet to get head pose*

We are applying the pose estimation algorithm in this snippet of code. This takes advantage of pose estimation using classical computer vision techniques explained in Chapter 2.



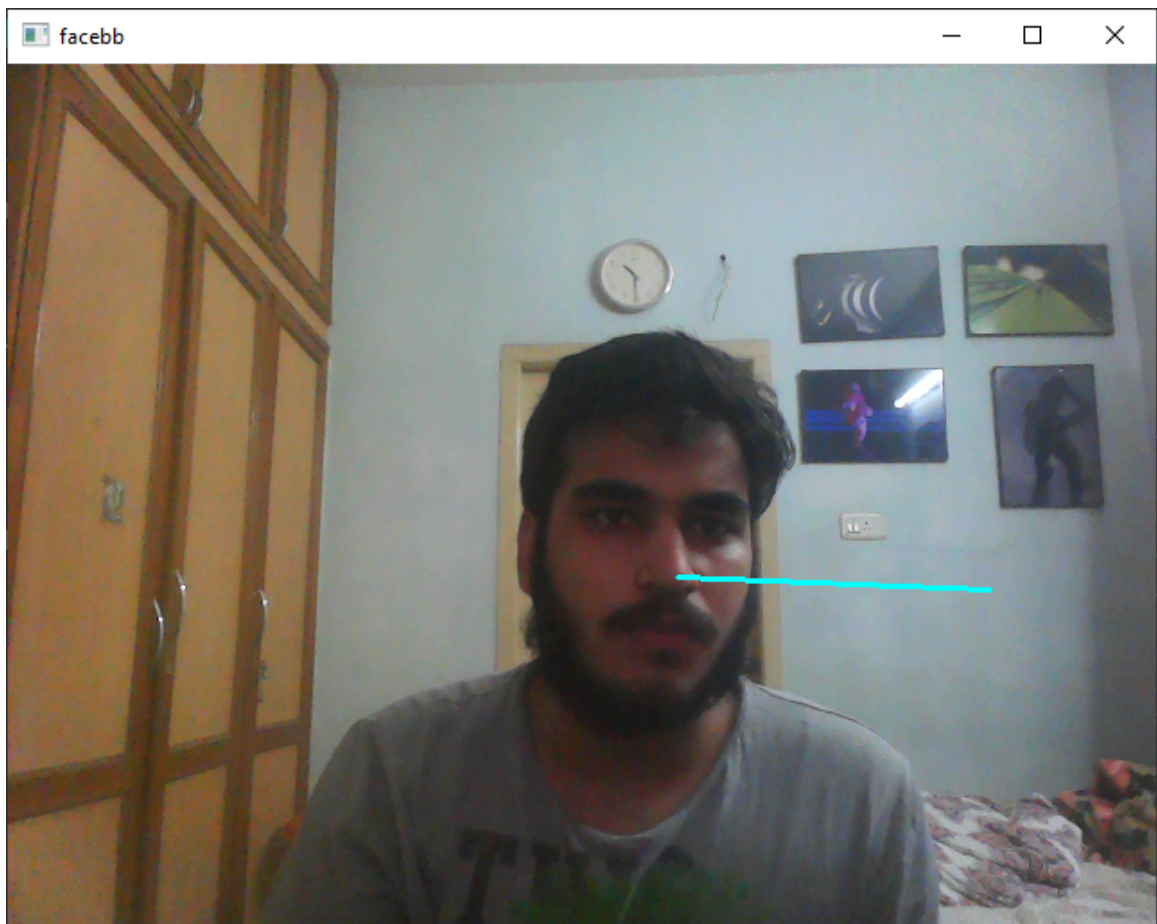
```
def get_head_pose(img, facial_landmarks):
    hp = [30, 8, 36, 45, 59, 55]
    size = img.shape
    image_points = np.array([(facial_landmarks.part(point).x, facial_landmarks.part(point).y) for point in hp], dtype="double")
    model_points = np.array([(0.0, 0.0, 0.0), (0.0, -330.0, -65.0), (-225.0, 170.0, -135.0), (225.0, 170.0, -135.0),
                             [(-150.0, -150.0, -125.0), (150.0, -150.0, -125.0)])

    focal_length = size[1]
    center = (size[1] / 2, size[0] / 2)
    camera_matrix = np.array([[focal_length, 0, center[0]], [0, focal_length, center[1]], [0, 0, 1]], dtype="double")
    dist_coeffs = np.zeros((4, 1))
    (success, rotation_vector, translation_vector) = cv2.solvePnP(model_points, image_points, camera_matrix, dist_coeffs)
    (nose_end_point2D, jacobian) = cv2.projectPoints(np.array([(0.0, 0.0, 1000.0)]), rotation_vector,
                                                    translation_vector, camera_matrix, dist_coeffs)

    p1 = (int(image_points[0][0]), int(image_points[0][1]))
    p2 = (int(nose_end_point2D[0][0]), int(nose_end_point2D[0][1]))
    return p1[0], p1[1], p2[0], p2[1]
```

*Fig 3.13 : Implementation of Head Pose Algorithm*

The outputs after applying the head pose estimation algorithm :



*Fig 3.14 Output of Head pose*

The output coordinates which have to be taken as input for the Neural Network.

```
3.2921204566955566
392 401 2101 498
3.383406639099121
391 401 494 402
3.4723269939422607
391 401 564 422
3.528273582458496
392 401 586 423
```

*Fig 3.15: Output Coordinates from algorithm*

The cropped eye image from the `get_eye_points` function is passed into the `ssd mobilenet` object detection algorithm. The 3 most probable bounding boxes of the pupils are returned as output, it amounts to 12 values (x,y,w,h for each bounding box). We are taking the 3 most

```
while True:
    _, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = detector(gray)
    key = cv2.waitKey(1)

    if (len(faces) >= 1):
        for face in faces:
            landmarks = predictor(gray, face)
            CC, angle = get_eye_points(frame, landmarks)
            x, y, w, h = get_face_points(face)
            x0, y0, x1, y1 = get_head_pose(gray, landmarks)
            a, b, c, d, e, f, g, h, i, j, k, l = main_pupil(CC)
            print(a, b, c, d, e, f, g, h, i, j, k, l)
```

*Fig 3.16: Snippet for finding the most probable coordinates*

Function to return 3 bounding boxes coordinates. It only returns the top 3 predictions after doing the conversions. The x and y coordinates are multiplied with 480 and 640 respectively as the image is split completely into pixels due to the YOLO(You only Look Once). We are finding only the top left and bottom right points as that will be enough to draw the rectangular bounding box.

```
def main_pupil(frame):
    image_np = np.array(frame)
    input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float32)
    detections = detect_pupil(input_tensor)
    num_detections = int(detections.pop('num_detections'))
    detections = {key: value[0, :num_detections].numpy()
                  for key, value in detections.items()}
    box = np.squeeze(detections['detection_boxes'])
    j = []
    for i in range(3):
        ymin = (int(box[i, 0] * 480))
        xmin = (int(box[i, 1] * 640))
        ymax = (int(box[i, 2] * 480))
        xmax = (int(box[i, 3] * 640))
        z = (xmin, ymin, xmax, ymax)
        j.append(z)
    return j[0][0], j[0][1], j[0][2], j[0][3], j[1][0], j[1][1], j[1][2], j[1][3], j[2][0], j[2][1], j[2][2], j[2][3]
```

*Fig 3.17: Snippet using YOLO algorithm*

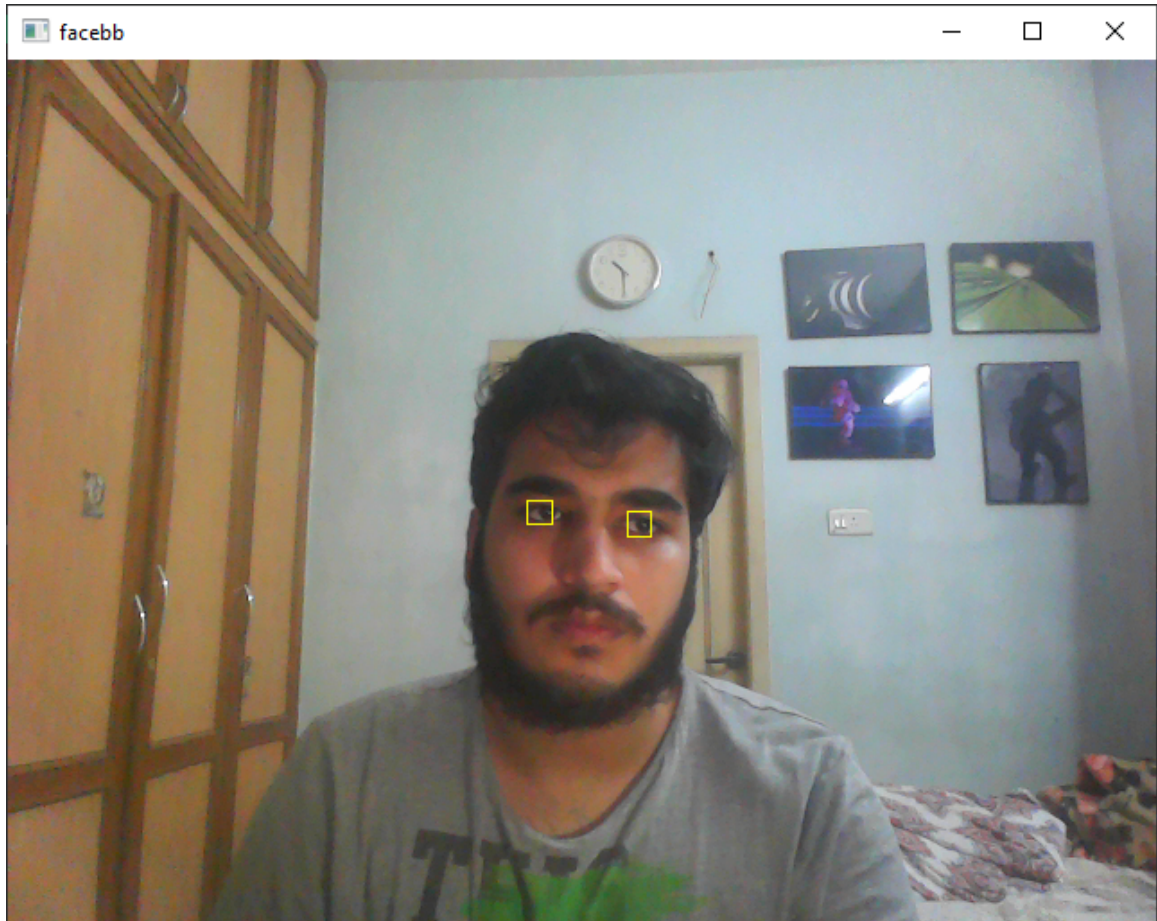
The previous function calls a sub function that makes the predictions.

```
def detect_pupil(image):
    image, shapes = detection_model.preprocess(image)
    prediction_dict = detection_model.predict(image, shapes)
    detections = detection_model.postprocess(prediction_dict, shapes)
    return detections
```

*Fig 3.18: Pupil Detection*

Output of the function will provide bounding boxes





*Fig 3.19: Output with Eye Bounding Box*

Output as Coordinates

```
27.011263608932495
400 328 414 341 329 325 344 339 401 322 416 337
27.162123203277588
401 330 415 341 330 328 343 339 403 322 416 337
27.311983823776245
333 334 346 344 401 335 415 346 393 337 405 345
27.51379680633545
```

*Fig 3.20: Output as coordinates*



The output after Labelling and Object Detection :



*Fig 3.21: Output after Labelling and Detection*

The recorded values appended in respective arrays are converted into numpy arrays. For labelling, the last array is either assigned as 0 or 1 depending upon whether the user is looking at the monitor or not.

```

A.append(a)
B.append(b)
C.append(c)
D.append(d)
E.append(e)
F.append(f)
G.append(g)
H.append(h)
I.append(i)
J.append(j)
K.append(k)
L.append(l)
ANGLE.append(angle)
X.append(x)
Y.append(y)
W.append(w)
HE.append(he)
X0.append(x0)
Y0.append(y0)
X1.append(x1)
Y1.append(y1)

```

*Fig 3.22: Appending values into arrays*

```

a[0] = np.asarray(A)
a[1] = np.asarray(B)
a[2] = np.asarray(C)
a[3] = np.asarray(D)
a[4] = np.asarray(E)
a[5] = np.asarray(F)
a[6] = np.asarray(G)
a[7] = np.asarray(H)
a[8] = np.asarray(I)
a[9] = np.asarray(J)
a[10] = np.asarray(K)
a[11] = np.asarray(L)
a[12] = np.asarray(ANGLE)
a[13] = np.asarray(X)
a[14] = np.asarray(Y)
a[15] = np.asarray(W)
a[16] = np.asarray(HE)
a[17] = np.asarray(X0)
a[18] = np.asarray(Y0)
a[19] = np.asarray(X1)
a[20] = np.asarray(Y1)
a[21] = np.asarray(0)

```

*Fig 3.23: Appending values into arrays(2)*

```

a.to_csv('data2.txt', sep=' ', header=None, index=False)
all_filenames = ['data{}.txt'.format(i) for i in range(1,3)]
combined_csv = pd.concat([pd.read_csv(f, sep=' ', header=None) for f in all_filenames ])
combined_csv.to_csv("data1.txt", index=False, encoding='utf-8-sig', sep=' ', header=None)

```

*Fig 3.24: Arrays are added in new text file which is merged*

The arrays are stored in a new txt file which is then merged with the previous recorded files.

```

df = pd.read_csv('data1.txt', header=None, sep=' ')
X = df.iloc[:, :21]
y = df.iloc[:, 21]

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=.3, random_state=666)

input = Input(21)
layer = Dense(21, activation='relu')(input)
layer = Dense(20, activation='relu')(layer)
layer = Dense(10, activation='relu')(layer)
layer = Dense(5, activation='relu')(layer)
output = Dense(1, activation='sigmoid')(layer)

model = Model(inputs=input, outputs=output)

checkpoint_name = 'Weights-{epoch:03d}--{val_accuracy:.5f}--{accuracy:.5f}.hdf5'
checkpoint = ModelCheckpoint(checkpoint_name, monitor='accuracy', save_best_only=True, mode='auto')
callbacks = [checkpoint]

model.compile(optimizer=Adam(), loss='binary_crossentropy', metrics=['accuracy'])

history = model.fit(x_train, y_train, batch_size=50, epochs=1000, callbacks=callbacks,
                    validation_data=(x_test, y_test), shuffle=True)

model.save('Model.h5')

```

*Fig 3.25: Features are given as input*

The recorded data is opened as a Dataframe. Here all the columns except the last are assigned X whereas the last column is assigned to y. Then the data is split into train and test in percentages of 70% and 30%. The train part is used for training while the test part is used for evaluation. After testing the model with multiple layers and multiple neurons, a 4 layer neural network with 20,10,5,1 neurons was adopted.

Here the input layer has 21 neurons because it is the number of features we have collected. Importantly, the output layer has only one neuron with a sigmoid activation function, because the resultant value should be between 0 and 1. The relu activation is the

best because if we have a lot of values then it will only take the positive values and the negative values are considered zero, because this makes calculation very easy and it is extremely fast as well. It makes computation for non-linear functions much simpler than what they actually are.

Then the model is compiled with an ADAM optimizer with the loss metric of binary cross entropy (because the desired output can only have 2 values).

Then the data is passed for model training with a batch size of 50 and 1000 epochs. The best weights are stored in a specific checkpoint using a callback function which can be called when the model is being deployed. Also the model is saved for future reference. There are a total of about 30000 data points.

The result of the last 5 epochs of model training :

```
Epoch 996/1000
297/297 [=====] - 0s 1ms/step - loss: 0.2162 - accuracy: 0.9167 - val_loss: 0.2645 - val_accuracy: 0.9014
Epoch 997/1000
297/297 [=====] - 0s 1ms/step - loss: 0.2195 - accuracy: 0.9141 - val_loss: 0.3068 - val_accuracy: 0.8862
Epoch 998/1000
297/297 [=====] - 0s 1ms/step - loss: 0.2211 - accuracy: 0.9136 - val_loss: 0.2620 - val_accuracy: 0.9033
Epoch 999/1000
297/297 [=====] - 0s 1ms/step - loss: 0.2185 - accuracy: 0.9144 - val_loss: 0.2564 - val_accuracy: 0.9052
Epoch 1000/1000
297/297 [=====] - 0s 1ms/step - loss: 0.2189 - accuracy: 0.9154 - val_loss: 0.2787 - val_accuracy: 0.8954
```

*Fig 3.26: Result of the last 5 epochs*

After training, the data is collected again and passed into the trained model for making predictions. The following code snippet has the full code for prediction par:

```

while True:
    _, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = detector(gray)
    key = cv2.waitKey(1)
    if (len(faces) >= 1):
        for face in faces:
            landmarks = predictor(gray, face)
            CC, angle = get_eye_points(frame, landmarks)
            x_y, w_he = get_face_points(face)
            x0_y0, x1_y1 = get_head_pose(gray, landmarks)
            a_b, c_d, e_f, g_h, i_j, k_l = main_pupil(CC)
            m = np.array([a_b, c_d, e_f, g_h, i_j, k_l, angle, x_y, w_he, x0_y0, x1_y1])
            m = tf.convert_to_tensor(np.expand_dims(m, 0), dtype=tf.float32)
            n = model.predict(m)
            image = cv2.rectangle(frame, (x_y), (x+w_y+he), (0,255,0), 5)
            image = cv2.rectangle(image, (a, b), (c,d), (0, 255, 255), 2)
            image = cv2.rectangle(image, (e, f), (g, h), (0, 255, 255), 2)
            if (n > 0.5):
                print('Yes')
                image = cv2.putText(image, 'Yes', (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 1, cv2.LINE_AA)
            else:
                print('No')
                image = cv2.putText(image, 'No', (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 1, cv2.LINE_AA)
            cv2.imshow('Prediction', image)

```

*Fig 3.27: Snippet for Prediction of output*

The only different thing is that the collected data is converted into tensors so that it is easy for the tensorflow model to accept it as input

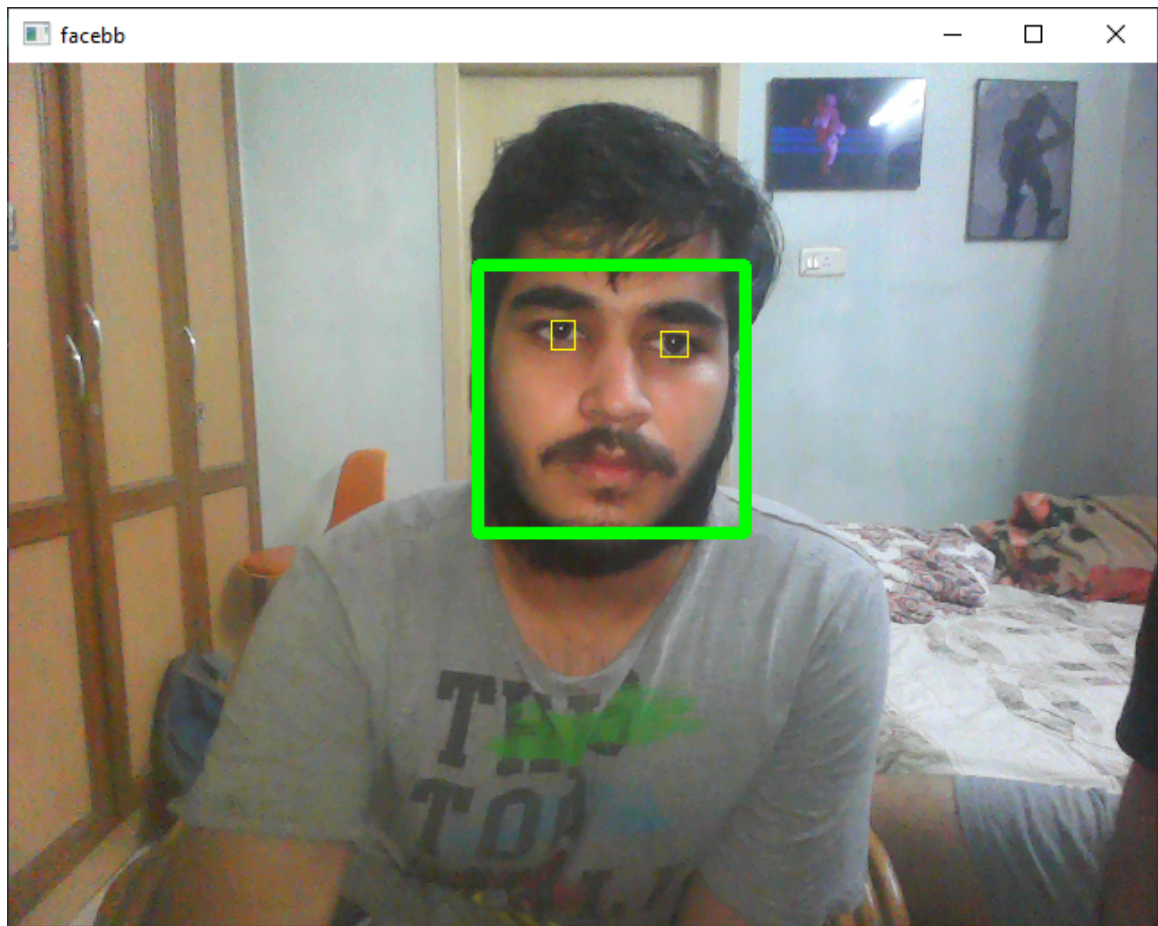
```

m = np.array([a_b, c_d, e_f, g_h, i_j, k_l, angle, x_y, w_he, x0_y0, x1_y1])
m = tf.convert_to_tensor(np.expand_dims(m, 0), dtype=tf.float32)
n = model.predict(m)
image = cv2.rectangle(frame, (x_y), (x+w_y+he), (0,255,0), 5)
image = cv2.rectangle(image, (a, b), (c,d), (0, 255, 255), 2)
image = cv2.rectangle(image, (e, f), (g, h), (0, 255, 255), 2)
if (n > 0.5):
    print('Yes')
    image = cv2.putText(image, 'Yes', (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 1, cv2.LINE_AA)
else:
    print('No')
    image = cv2.putText(image, 'No', (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 1, cv2.LINE_AA)
cv2.imshow('Prediction', image)

```

*Fig 3.28: Conversion to Tensors*

FINAL OUTPUT :



*Fig 3.29 : Final Output*

## **CHAPTER 4**

### **CONCLUSION**

#### **4.1 CHALLENGES AND CONSTRAINTS FACED**

- This project does not take into account an additional face when it is presented with two faces, it recognizes only one face and tracks it.
- There is a very mild lag between the output seen on the screen and the live movement of the user. While we have minimised the lag by using Dlib and MobileNet SSD model for having faster speed, a minute lag is unavoidable.
- The accuracy of the project was not good until we started implementing the head pose estimation algorithm. It was very challenging to understand that algorithm and actually apply it.
- The accuracy right now is just over 90% and as the project keeps learning it will be more accurate in nature.
- The project is incapable of finding the difference between a real human and a life size cardboard of a human face.



## 4.2 FUTURE IMPLEMENTATIONS

The project is a Student Logger using Gaze Tracking but what it essentially does is to detect a face, find the facial coordinates, the face area, the centre of the pupil and the eye coordinates and head tilt angle to give as input into the 4 Layer Neural Network and check where the gaze of the user is in real time. This project has been designed in such a way that it runs at a very high speed on low end devices as well and has no hardware requirements apart from a basic webcam.

This project can be picked by institutions or companies that would like to know if the students are paying attention and also analyse when the attention of students peak and what not. It can be used in the field of psychology to study the attention span of kids to certain visuals and through that finding ways to make learning more interesting through the internet.

This technology can be applied in various fields and it may end up revolutionizing the industry as there have not been any gaze tracking models that can be run with such high speed and accuracy without any costly requirements.

For instance, the road safety can be increased by live checking whether the driver is looking at the road or not, if he looks away or down at his phone then it will be recorded. We can also use it for streaming platforms by making the video pause when someone is not looking at the screen and has to look away, what can also be another use is to get data on which are the movies that make their people have their eyes glued to the screen. This can largely benefit the platform as they can make their recommendation algorithm give a push to the movies that will be most liked by people. This feature can be used in any surveillance industry to a very great degree because of its nature to be quick and work with low resolution images as well.



### 4.3 CONCLUSION

Education is an integral part of anyone's life and while we do value it now, it can seem a little boring when you are young, especially when you are attending classes through a device that you usually use to watch fun videos and play games on. Hence it becomes essential to have a check on whether they are paying attention or not. This project has succeeded in creating a way through which we can check whether the kids are paying attention to the class or not.

Through using the webcam on the system that the child uses to attend class, we track his gaze 6 times every second for bringing the effect of live video-like tracking. We use these images to extract features like face area, face bounding box coordinates, pupil bounding box coordinates, head pose, center of pupil and then pass the value into the 4 Layered Neural Network that we have created. We take out the sigmoid value and use it to find out whether the user is looking at the screen or not. This project has a trained dataset with over thirty thousand data points and this has helped in recording a very good accuracy rate of 90% as well.

## REFERENCES

[1]Gaze and Eye Tracking : Techniques and Application in ADAS

Authors : Muhammad Qasim Khan,

Sukhan Lee

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6960643/>

[2]A Survey On Eye Gaze Tracking Techniques

Authors: H.R.Chennamma

Xiaohui Yuan

<https://arxiv.org/ftp/arxiv/papers/1312/1312.6410.pdf>

[3] NVGaze : Anatomically informed Dataset for Low Latency, Near Eye Gaze Estimation

Authors: Joohwan Kim

Michael Stengel

Alexander Majercik

Shalini De Mello

David Dunn

Samuli Laine

Morgan McGuire

David Luebke

[https://research.nvidia.com/publication/2019-05\\_NVGaze%3A-An-Anatomically-Informed](https://research.nvidia.com/publication/2019-05_NVGaze%3A-An-Anatomically-Informed)

[4]Light-weight Head Pose Invariant Gaze Tracking

Authors: Rajeev Ranjan

Shalini De Mello

Jan Kautz

[https://research.nvidia.com/sites/default/files/pubs/2018-06\\_Light-weight-Head-Pose/lightweight-gaze.pdf](https://research.nvidia.com/sites/default/files/pubs/2018-06_Light-weight-Head-Pose/lightweight-gaze.pdf)

[5] A Review And Analysis Of Eye Gaze Estimation systems, Algorithms and Performance Evaluation Methods in Consumer Platforms

Author : Anuradha Kar

<https://arxiv.org/ftp/arxiv/papers/1708/1708.01817.pdf>

[6] Deep Multitask Gaze Estimation With A Landmark- Gaze Model

Authors: Yuyu

Gang Liu

Jean-Marc Odobez

[https://openaccess.thecvf.com/content\\_ECCVW\\_2018/papers/11130/Yu\\_Deep\\_Multitask\\_Gaze\\_Estimation\\_with\\_a\\_Constrained\\_Landmark-Gaze\\_Model\\_ECCVW\\_2018\\_paper.pdf](https://openaccess.thecvf.com/content_ECCVW_2018/papers/11130/Yu_Deep_Multitask_Gaze_Estimation_with_a_Constrained_Landmark-Gaze_Model_ECCVW_2018_paper.pdf)

[7] NVGaze: An Anatomically Informed Dataset For Low- Latency, Near Eye Gaze Estimation

Authors: Joohwan Kim

Shalini De Mello

Michael Stengel

Alexander Majercik

David Dunn

Samuli Laine

Morgan McGuire

David Luebke

[8] [https://github.com/tensorflow/models/blob/master/research/object\\_detection/models/bi\\_directional\\_feature\\_pyramid\\_generators\\_tf2\\_test.py](https://github.com/tensorflow/models/blob/master/research/object_detection/models/bi_directional_feature_pyramid_generators_tf2_test.py)

[9] [https://github.com/tensorflow/models/blob/master/research/object\\_detection/models/center\\_net\\_mobilenet\\_v2\\_fpn\\_feature\\_extractor.py](https://github.com/tensorflow/models/blob/master/research/object_detection/models/center_net_mobilenet_v2_fpn_feature_extractor.py)

[10] [https://github.com/tensorflow/models/tree/master/research/object\\_detection/models](https://github.com/tensorflow/models/tree/master/research/object_detection/models)

[11] <https://learnopencv.com/head-pose-estimation-using-opencv-and-dlib/amp/>

[12]

<https://towardsdatascience.com/real-time-head-pose-estimation-in-python-e52db1bc606a>

[13] <https://technostacks.com/blog/yolo-vs-ssd>

[14]

<https://deeptai.org/machine-learning-glossary-and-terms/hidden-layer-machine-learning>

[15] <https://developer.ibm.com/technologies/artificial-intelligence/articles/cc-machine-learning-deep-learning-architectures/#:~:text=Because%20deep%20learning%20relies%20on>

[.build%20these%20deep%20learning%20structures.&text=Artificial%20neural%20network%20\(ANN\)%20is,underlying%20architecture%20behind%20deep%20learning.](#)

[16]

[https://www.sas.com/en\\_in/insights/analytics/neural-networks.html](#)

[17] [https://www.guide2research.com/research/online-education-statistics](#)

[18]

[https://www.acetutoring.com/blogs/2020/9/17/battling-against-student-attention-span-in-online-learning](#)

[19] [https://www.sciencedirect.com/science/article/pii/S1877705817326620](#)

[20] [https://learnopencv.com/wp-content/uploads/2016/09/ImageFormationEquation.jpg](#)

[21]

[https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5](#)