Online Engage-Measurement in Tutoring Session

by

Saadman Omar Siddique
19301180
M. Shafiul Alam
19301194
Mahmud Alam
19301214
Nabil Hasan
19301222
M. M. Hasan Tajwar
19301240

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
January 2023

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

[illegible]

M. Shafiul Alam
19301194

Mahmud

Mahmud Alam
19301214

Nabil Hasan

Nabil Hasan
19301222

M. M. Hasan Tajwar

M. M. Hasan Tajwar
19301240

Saadman Omar Siddique

Saadman Omar Siddique
19301180

Approval

The thesis/project titled “Online Engage-Measurement in Tutoring Session” submitted by

1. Mahmud Alam (19301194)
2. Nabil Hasan (19301222)
3. Shafiul Alam (19301194)
4. Saadman Omar Siddique (19301180)
5. M.M.Hasan Tajwar (19301240)

Of Spring, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 19, 2023.

Examining Committee:

Supervisor:
(Member)

[illegible]

Dr. Md. Khalilur Rhaman
Associate Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Ethics Statement

In doing research for this thesis, we commit to observe the highest ethical standards by obtaining informed consent from all participants, maintaining the confidentiality of sensitive information, and conducting our work with integrity and neutrality.

Abstract

The COVID-19 pandemic has brought about a significant change in the way education is delivered worldwide. Restrictions have forced schools, colleges, and universities to hold classes online using video communication services. While this method of teaching has its advantages, one major challenge is determining student engagement during virtual sessions. In traditional classrooms, it is easier to observe student's interest and engagement through body language and movements. However, this is not the case in an online setting, where monitoring engagement requires more resources. To address this issue, we have undertaken research to develop a system called "Online Engage-Measurement" that automates the process of monitoring engagement by measuring attention and detecting screen sharing. This system will be faster, more efficient, and accessible to educators everywhere. It uses screen sharing detection, face recognition, head position, and eye gaze estimation, as well as an algorithm called "AttentionEstimator" to determine engagement levels. The system detects the attentiveness of both students and teachers and generates a report for analysis. Besides, our research is unique as this field has not yet been implemented, and our system is the result of our research contributions, which will help us to be a part of the Fourth Industrial Revolution. This initiative has the potential to improve the future of education and solve many problems, such as the development of proctor-less examination system. Utilizing such attention measuring system in online education can provide valuable insights for educators to adapt and refine their teaching methods to align with the needs of their students. It allows for the assessment of the effectiveness of instruction and detection of areas for improvement in student performance, thus providing valuable information to enhance the educational experience for students. Thus, the system we have built has the potential to improve student learning experiences and boost tutoring session efficiency.

Keywords: Online Engage-Measurement, Screen Sharing detection, Face Recognition, Head pos, Eye Gaze Estimation, AttentionEstimator, System, Proctor-less, Attentiveness, Unique

Dedication

We dedicate this research to our parents and to the future of online education.

Acknowledgement

Firstly, all praise is due to the Great Allah, for whom our thesis has been completed without any major interruption.
Secondly, to our Advisor Dr. Md. Khalilur Rhaman, who had given us the incentive to start our thesis on this topic, for his kind support and advice in our work. He helped us whenever we needed help.
And finally to our parents, without their throughout support it would not be possible. With their kind support and prayers, we are now on the verge of our graduation.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Nowadays, live internet teaching is a popular way of education. The accessibility and availability of joining a class online from home or any other location with easy-to-access internet and internet technologies, in addition to the current covid-19 pandemic, have all pushed the popularity of online classes sky-high. With that regard, many concerns have also risen that were either absent or not so prevalent in the counterpart of online classes, such as the absence of attention and participation of the participants of the class like the in-person classes. This becomes an issue when the participant is either a student's parents or the authorities of the educational institution who are concerned regarding the quality and effectiveness of the class. In this case, an automated technique for monitoring class participants' involvement or attention becomes essential. With this purpose in mind, we propose an engagement-measurement system that will automate the majority of this procedure and monitor both the student and teacher's attention levels throughout a 1-1 online tutoring session.

1.2 Aims and Objectives

We aim to build such a system that will attempt to measure the engagement level of both the student and tutor in a 1on1 online session. We will detect the screen sharing of the participants, including the facial features which include mainly the head pose and the eye gaze, and in the end, generate a report that will allow us to determine the engagement and attentiveness of the parties involved.

1.3 Research Methodology

First, we have organized our working method into modules so that we may work on each component individually. For our first module, screen sharing, we utilized the OpenCV library to map the screen's contour at any given time in order to determine if the screen is being shared and to identify the individual sharing the screen. The second module of our system is facial feature recognition, which is utilized to recognize faces in the video input. For face detection, we utilized the 'face_recognition' library to find the faces in the video that was inputted. Then we used the 'K-means' algorithm to cluster the detected faces into several folders. For the third module, in the head position estimation part, the 6DRepNet model was utilized to estimate the position of the detected faces. Using the 'MediaPipe' library, we monitored the participants' eye movements to estimate their gaze direction in the eye gaze estimation part. These modules were utilized for evaluating students' or tutors' attentiveness in the final module which is the attention measurement module. We also utilized Python libraries for graph production such as matplotlib, seaborn, NumPy, and pandas to visualize our data for the result.

1.4 Thesis Orientation

At the start of chapter 2, we covered related work in the face recognition and attention estimation domains. In the third chapter, we then present an overview of our system. Individually, we analyze and discuss each module of our system. We discussed the data we utilized. Finally, we discuss how each of these components contributes to the intended outcome. In chapter 4, the implementation phase is discussed. In chapter 5, we discuss our findings and the precision of our model.

Chapter 2

Literature Review

Attention detection is a well-researched topic. It is essential now because almost everything is online-based. From educational purposes to international conferences, everything is being conducted online. As it is becoming a trend to conduct online sessions, especially for educational purposes, administrators are now seeking applications that will help them determine their total attentiveness. As said in [18], teachers often rely on detecting and responding to overt student behaviors as proof of their attention in face-to-face settings. However, in an online session, teachers may only be able to see a student's head and shoulders, limiting the amount of information provided. Even if the information is recorded by the instructor, the instructor's teaching proficiency is not documented in any way. We can determine attentiveness manually. However, that is a lot of work. Therefore, automation of attention detection is a must.

2.1 Related Works

The researchers [3] combined the YOLO (You Only Look Once) algorithm with the VGG16 CNN to make a face detection system. The system was able to detect the faces faster than previously tested systems. The method was divided into three stages; creating the truth database, feature extraction, and face detection. In the feature extraction part, a pre-trained VGG16 model with 32 layers was used. Some of the layers used were the convolutional layer, rectified linear unit layer (ReLU), pooling layer, fully connected layer, dropout layer, and classification layer. At the end of the face detection stage, the output obtained from the pre-trained VGG16 was combined with the YOLOv2 algorithm. As the name suggests, the "you only look once" YOLO algorithm needs to process each image just once to detect all the features, so it is very fast. During the experiment, the system was able to detect the test image set with 95% precision. Other works utilizing the VGG16 model include using a pre-trained VGG16 model in conjunction with a Deep Convolutional Neural Network to classify images using transfer learning, as shown in . Transfer learning is applying previously learned knowledge to solve more modern problems faster and with better solutions. Here, the CNN model was developed in Python using libraries like TensorFlow and Keras. Firstly, the images are passed through layers of the CNN model, and the output is used to classify images. The output images obtained are fine-tuned, and the VGG16 model is implemented to classify the images, check the accuracy, and validate the data. The layers used in this VGG16 model were similar to the layers of the VGG16 model mentioned in the previous paragraph. The dataset used here comprised over 25000 images of various dogs and cats, and the system implemented was to distinguish the two animals' images accordingly. After fine-tuning the images and using the VGG16 model, they were able to obtain an accuracy of 95.4%.

Lindelof, A & Eriksson, J (2015) said in [9], that a better approach to student attention detection is the MB-LBP (multi-block local binary pattern), which required less training time than the V-J algorithm and also yielded more accuracy. They defined attention in similar ways as [8]. Although they concluded that both the V-J and MB-LBP methods yielded almost as accurate 10 results, they admitted that this could be further improved with better training environments

The research paper [22] presented a "Help-Me-Watch" (HMW) system that used AI algorithms to provide personalized video summaries of online classes delivered via Zoom. It assists in resolving the drawbacks of online learning and assists students with class reviews by recommending these classes to students based on all attendees' attention levels recorded during live sessions. The HMW software, which is downloaded onto the student's laptop and is based on a real-time eye blink detection algorithm that computes the eye aspect ratio (EAR) between the height and width of the eye, estimates the student's attention level. The basic video summary technique employs 1-minute aggregations of attention levels, and it eliminates the attention levels of students who came late, departed early, or were not looking at their screens; the missing portion is then included in their video summary. The mean attention level and volatility, which is the degree of variation over time, which is often quantified by the standard deviation of logarithmic changes in attention levels of participating students, are then merged and outputted into tables and graphs. Figure 3 of the proposed paper demonstrates that a class lasted 48 minutes, that 12 students utilized the HMW system, and that the stacked bar chart provides anonymized aggregated attention levels from those 12 students, outputting an average attention level of 78%.

In [15], they first categorized engagement detection into three different methods, automatic, semi-automatic, and manual. As we focus on the automatic side it is further broken down into Computer Vision-based detection, sensor data analysis, and log file analysis. For computer vision-based detection systems, the video feed is first captured using cameras. The system detects Regions of Interest from the video feed and uses modules to extract, classify, track and finally give a decision regarding the percentage of engagement. Computer vision-based detection mainly focuses on the detection of facial expressions, gestures and posture, and eye movement. For facial expression detection, two methods are used, part-based and appearance-based. For appearance-based regions from the whole face are used to generate patterns which are then detected and classified by feature extraction techniques such as Local Binary Patterns (LBP) and Deep Multi-instance learning (DMIL). For gesture and posture detection, the gestures were captured using cameras and graph-based visual saliency (GBVS) was used to detect potential deictic gesture regions, and Support Vector machines (SVM) based classifiers were used to determine the accurate gestures from the detected regions.

Eye tracking is very important in terms of measuring attentiveness. Over the years, people have used many different eye-tracking systems. [12] One of them is the Tobii EyeX eye-tracking controller. The pupil center and corneal reflection technique are used in the design of these eye trackers. We can calculate the position of the pupil based on the position of an infrared light glint on the cornea. To improve the accuracy of gaze point estimation, the Tobii EyeX Engine offers a native calibration method (TNC) that must be completed before using the engine with a new user. The technique is necessary to determine the setup's geometry (e.g., screen size, distance, etc.) and to gather data on the subject's corneas' light refraction and reflection characteristics. For the experiment, in order to reliably measure the accuracy and precision, observers were positioned at $\approx 700mm$ from a computer monitor with the head stabilized by a chin and forehead rest. The individuals went through the calibration and testing methods specified in the "Calibration Technique" for monocular and binocular vision. The tests were carried out on a PC equipped with an Intel 12 Core i7-4700 CPU @ 2.40 GHz and 12 GB of RAM, connected to a 17-inch LCD with 1920 x 1080 resolution at 60 Hz, and running Windows 7 Professional.

Research Paper [6] presents us with different types of head pose estimation methods. Some examples are Appearance Template Methods, Detector Array Methods, Nonlinear Regression Methods, Manifold Embedding Methods, Flexible Methods, Geometric Methods, Tracking Methods, and Hybrid Methods. It is a very exciting field with a room full of improvement, people desire off-the-self, universal head pose estimation applications for their work. Estimating head posture is an obvious next step in bridging the information gap between humans and computers. This basic human skill gives a lot of information about people's intentions, motivations, and attention in the world. Systems that can better interact with humans can be constructed by simulating this skill. Most head position estimation methods work from the viewpoint of a rigid model, which has its own set of restrictions. The complexity of developing head posture estimation algorithms arises from the variety of individual appearances, as well as variances in illumination, background, and camera geometry. Murphy-Chutorian & Trivedi (2008) proposed design criteria as a guide for future development. The following design criteria include Accurate, Monocular, Autonomous, Multiperson, Identity & Lighting Invariant, Full Range of Head Motion, Resolution Independent, and Real-Time. It seems that no single system has met all these criteria.

The study [4] is focused on a scheme that uses contour based features to detect objects categorically. For object detection for images on screen contour mapping has many advantages of traditional texture mapping as it can match image structures with large spatial extents while remaining largely invariant to lighting conditions and object color.The paper in [1] talks about how a modified K-means algorithm can be used to cluster data. The modified algorithm implements measurement of distance on the idea of point symmetry which can be applied into human face detection and data clustering. The study in [11] talks about how a dedicated eye tracker can be used to collect eye gaze data to analyze loss of attention while doing jobs that required the use of computers. The behavior of eye patterns were observed and was found that patterns varied distinctly when attention was not being paid.The research in [24] proposes a webcam-based eye gaze detection method which allows free head movement of the user. Latest Deep learning techniques such as MediaPipe were used so that the method functions in real time with live video feed while maintaining acceptable frame rates which is quite necessary in real time processing. The paper in [5] talks about how paying attention plays a part in the learning capability of students. It talks about how the wandering of mind of students during classes affects their focus and their overall learning. The paper in [13] talks about a multi loss convolutional neural network which was trained on the 300W-LP dataset. The neural network predicts intrinsic Euler Angles yaw,pitch and roll directly from intensity of images through regression and joint based pose classification which can predict the headpose of the person more robustly and elegantly.

In paper [17], they investigate methods for estimating students' engagement levels based on a set of facial and head movement behaviors that describe dynamic movement of the eye, head, and mouth. The findings show plausible nonlinear correlations with EEG-based attention measurement, laying the groundwork for future research into methods to fuse information from those two modalities.In study [20], the researchers tracked attention using biometrics and machine learning. In particular, they concentrate on discussions and analyses of techniques that make use of eye gaze, facial expressions, body movements, behavioral biometrics like brainwave analysis, and multimodal biometrics. They conclude by talking about some exciting new directions for future research, with a focus on multimodal biometric methods. The study [16] describes how eye-tracking data can be used to forecast a student's attention as a gauge of their emotional state throughout a lesson. Using the Extreme Gradient Boosting machine learning technique, accuracy of 77% was achieved in this study. The results show that eyegaze can in fact be a foundation for creating a predictive model.

Chapter 3

Methodology

3.1 Workflow

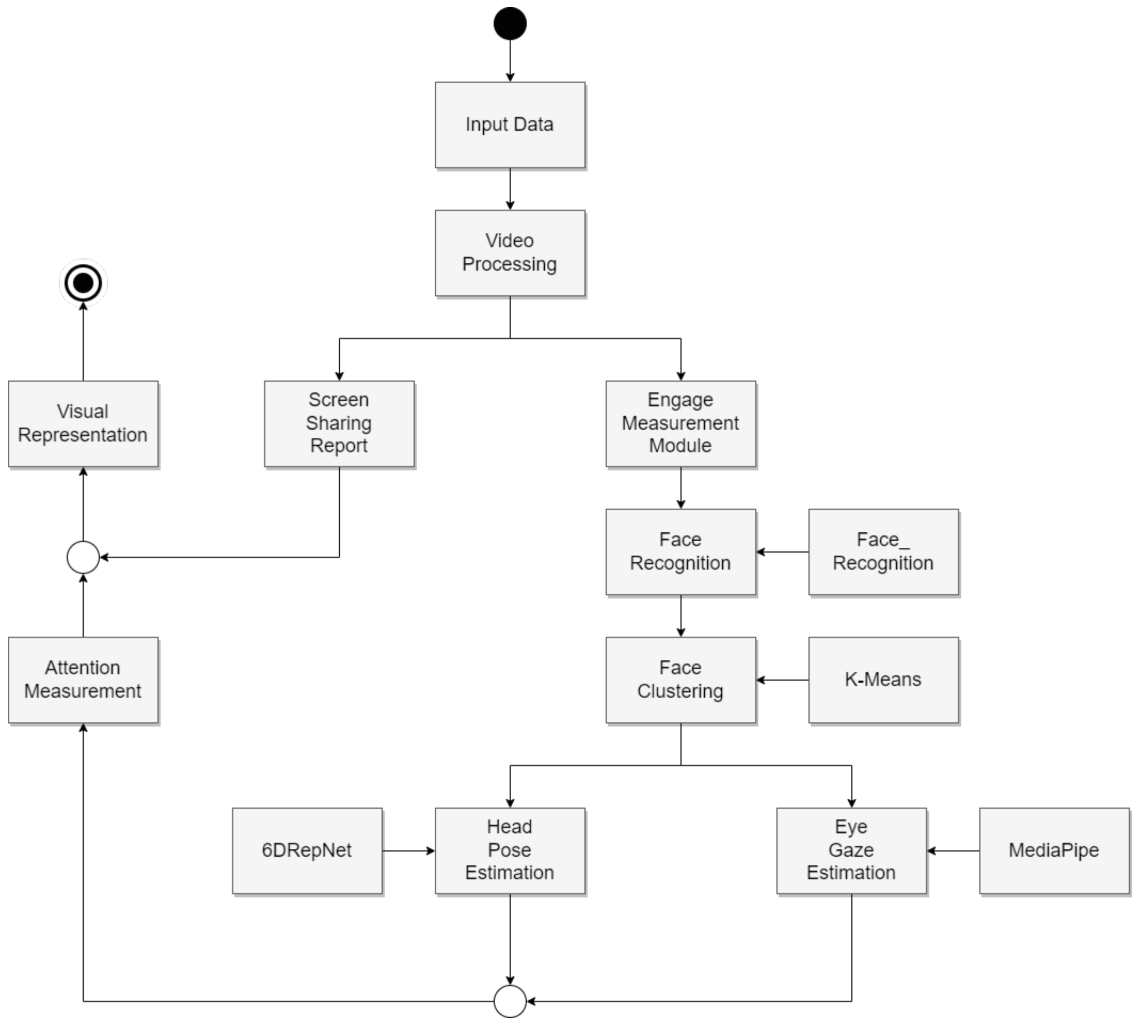


Figure 3.1: Workflow of OEMTS

In our workflow diagram, we begin with Input Data, which is then processed through the Video Processing module. This module is divided into two sections: the Screen Sharing Report and the Engagement Measurement Module.

The Engagement Measurement Module utilizes the 'face_recognition' library to perform Face Recognition on the input data. The output of this step is then fed into the Face Clustering module, which employs the K-Means library for grouping similar faces together. The Face Clustering step splits into two sub-parts: Head Pose Estimation and Eye Gaze Estimation. The Head Pose Estimation sub-module uses the 6DRepNet model, and the Eye Gaze Estimation sub-module uses the MediaPipe library. These two sub-parts are then integrated into the Attention Measurement module, which measures the attentiveness of the individual.

The Screen Sharing Report and Attention Measurement are then used to generate a Visual Representation of the input data, which provides insights into the engagement and attention levels of the participants in the video.

This workflow is designed for providing engagement and attention measurement during the video conference or screen-sharing scenario, which enables us to have a more efficient and productive remote working or studying environment.

3.2 Dataset

The dataset used in this study consisted of video recordings of 1-on-1 sessions, mainly zoom sessions. These recordings were captured using in-built screen recorders of the respective software and recording software such as 'OBS Studio' on laptops and computers using webcams under different lighting conditions. The total number of video files in the dataset was four, with an average duration of 45 seconds per video. For testing purposes, these videos were trimmed to achieve a smaller video duration. The accuracy of the results of these modules was evaluated manually. While testing the system, a single video from the dataset was inputted into it at a time.

Therefore, four video files were used to evaluate the performance of our system. The following features were observed for each video:

Video 1: A video session featuring two individuals with their webcams turned on and screen sharing active throughout the duration of the video.
Video 2: A tutoring session featuring one individual with their webcam turned on and screen sharing active throughout the duration of the video.
Video 3: A tutoring session featuring two individuals with their webcams turned on, but no screen sharing present.
Video 4: A tutoring session featuring two individuals, but neither had their webcams turned on, with screen sharing active at certain points during the video.

The first step in processing the dataset was to detect and extract the faces from each frame of the video. This was accomplished using the Python 'face_detection' library. The extracted face images were then resized to a specific resolution of 100x100 pixels.

Next, the extracted face images were grouped and sorted into different folders depending on the recognized individual in each image. This was accomplished by utilizing the 'KMeans' algorithm.

The clustered face images were then used as input for various modules, including head pose estimation and eye gaze position. Head pose estimation was performed using the 6DRepNet model, while eye gaze position was determined using the 'MediaPipe' library. The accuracy of the results of these modules was evaluated manually.

Altogether, the collection of videos utilized by our system was compiled from a variety of sources, including class recordings from our university, YouTube, and our own recordings. Overall, the dataset played a crucial role in the success of the proposed system and its ability to accurately estimate head pose and eye gaze position, screen sharing duration and the attentiveness of the student and the tutor.

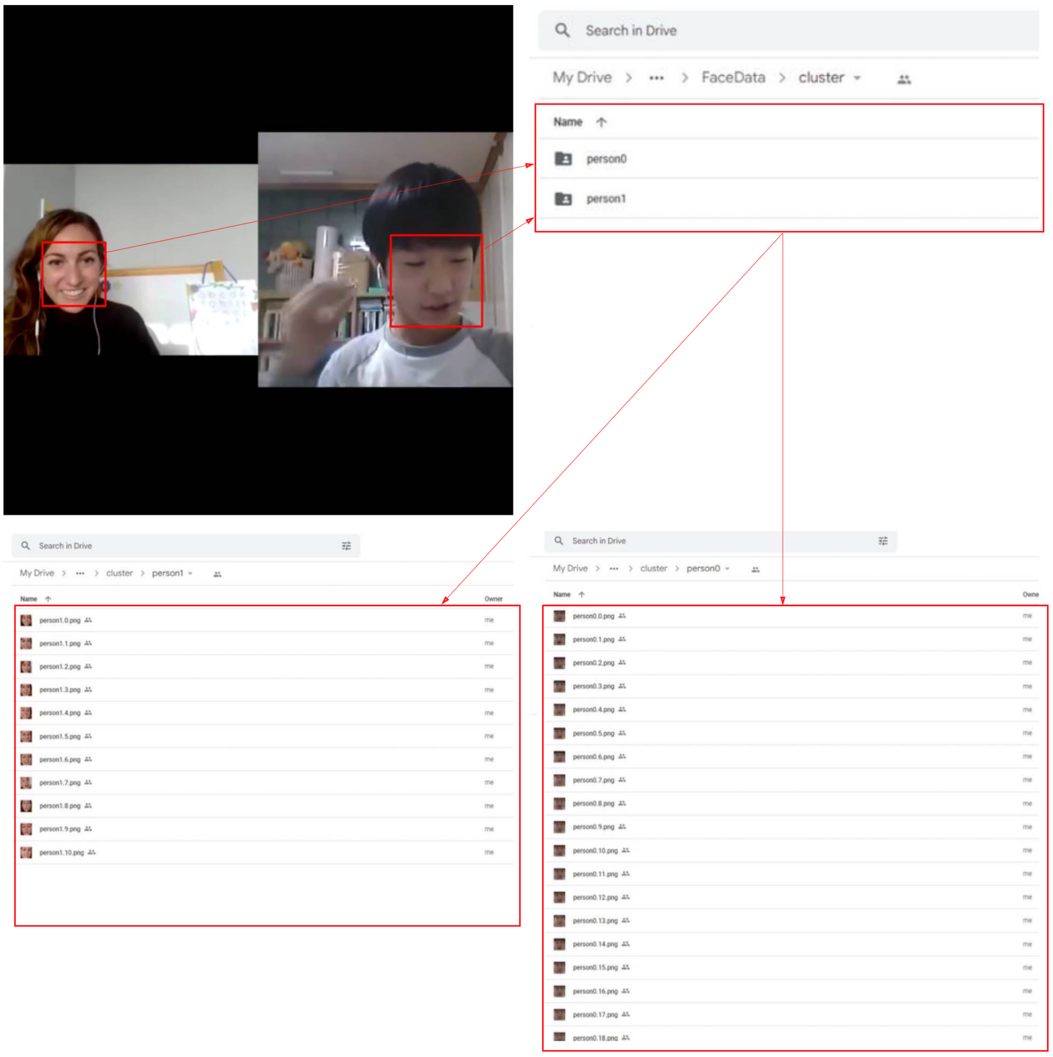


Figure 3.2: Images of faces extracted from video

3.3 Model

For our system, we employed a variety of libraries and models to develop our facial recognition system. We used the 'face_recognition' library to detect and extract facial features from images and videos. The 'scikit-learn' library was used to implement the KMeans algorithm for clustering the detected faces. We also used 'NumPy' and 'OpenCV libraries for image processing and manipulation. The 'Mediapipe' library was used for eye position estimation. For head pose estimation, we used the '6DRepNet' model, which is a deep learning model. The model is trained on a large dataset of faces and is able to extract a high-dimensional representation of a face, allowing for accurate recognition and tracking of individuals. We created our own algorithm for estimating the attentiveness of an individual in a session. Overall, our system presents a practical and efficient approach for estimating the engagement of an individual in a video input. We used a combination of computer vision techniques and machine learning models to develop our system to assist online tutoring sessions.

3.3.1 Screen Share Detection

One of the features of our system is to detect if the screen is being shared or not. By measuring the amount of time the screen is being shared, we can have an idea of how long the participants were active in the meeting. In order to detect if the screen is being shared, we used Contour Detection using the OpenCV library.

Contour Detection

Contours are boundary pixels of objects that have the same intensity and color. By using OpenCV we can find contours from images. Firstly, we read the video frame by frame. We then resize the frame so that we can work with the frames more conveniently. After that, we grayscale the images obtained from the frame so that it becomes easier to process. Then, we use the 'cv2.findContours()' function to find the contours of the images that we obtained from the frames. The function takes an image and returns a list of contours which are each represented by a list of points. Contours are organized by hierarchy, arranging the outermost contour at the beginning of the list and the innermost contours nested inside of it. After detecting the contours, we find the contour count of the frame by using the 'len(contours)' function. Lastly, we set a threshold of 100 contour counts to determine if a screen is being shared. If the contour count is above the threshold, then we can evaluate that there are various objects present on the screen and thus that means the screen is being shared. On the other hand, if the contour count is below the threshold, then we can estimate that there are fewer objects present on the screen and the screen is mostly black, and no screen is being shared.

3.3.2 Face Recognition and Clustering

The 'face_recognition' python library is an effective instrument for detecting and identifying faces in video and images. This library utilizes sophisticated algorithms to recognize and extract facial features [10], such as eyes, nose, mouth, and face contours, from images. It also uses deep learning algorithms to identify faces and match them against a database of known faces. The library employs dlib's state-of-the-art deep learning-based facial recognition. It is capable of comparing a face in an image to a set of known faces and returning the name of the individual with the highest degree of resemblance. Additionally, it allows for the identification and recognition of several faces inside a single image or video frame, which is ideal for our system. Thus, our approach leveraged the power of the 'face_recognition' python

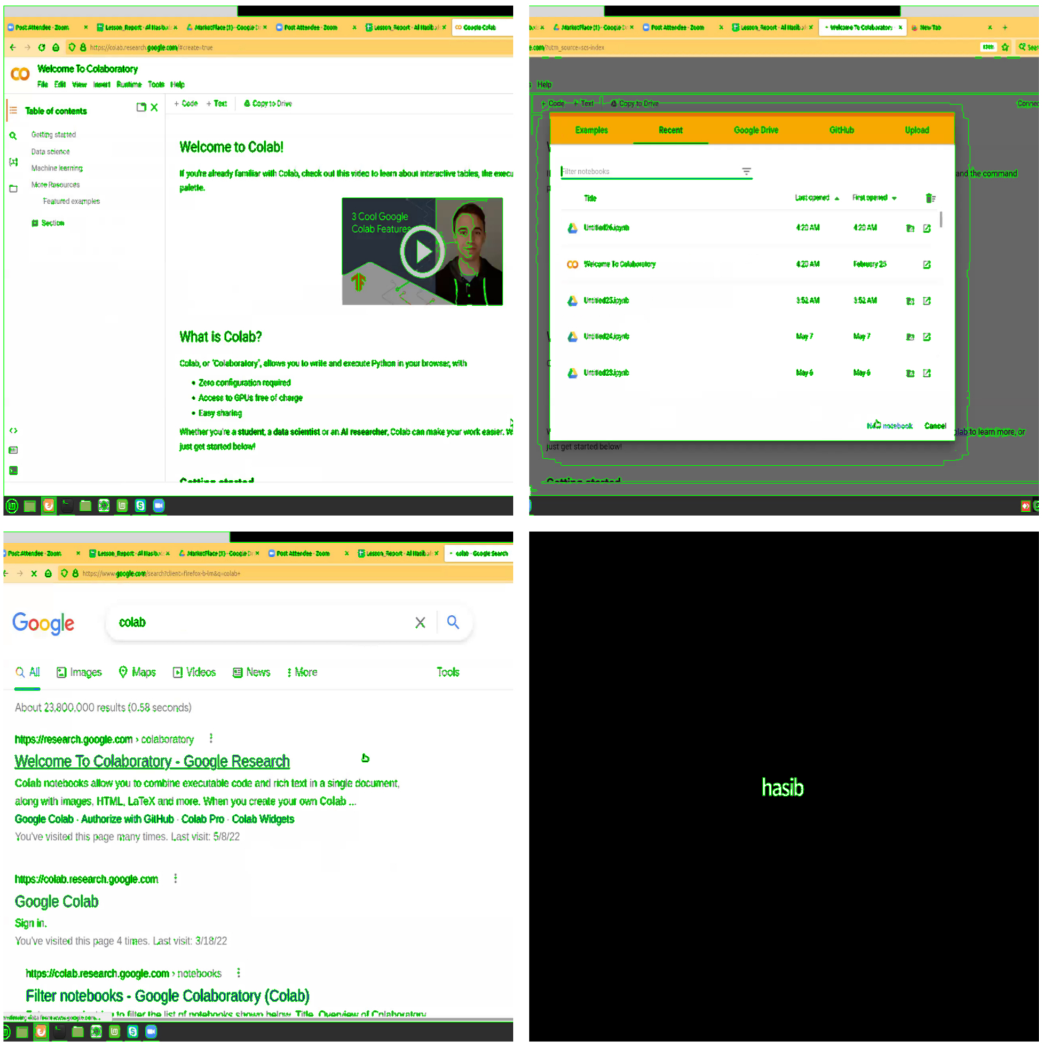


Figure 3.3: Contour Detection on Screen

library to detect and extract faces from the frames of the video that was inputted into the system. When an image or a video frame is processed, the deep learning model scans the image and detects the presence of faces. Once a face is detected, the model then extracts a set of facial features, including the coordinates of the eyes, nose, mouth, and face contours. These features are then used to create a unique facial descriptor for each face, which serves as a numerical representation of the face. We resized the detected faces in a specific resolution, saving them using a loop, and named them in a specific format.

The detected faces were then processed and analyzed through the use of the 'KMeans' library for clustering [2]. KMeans is a well-known algorithm that uses 'Scikit learn library' for clustering and categorizing data based on their similarities. It is an unsupervised machine-learning technique that divides a batch of data points into a predetermined number of clusters. The clustering process enabled us to group the detected faces according to their similarities, which allowed us to categorize them into subfolders for further analysis. The clustering process begins by initializing the cluster centroids randomly. The algorithm then assigns each detected face to the cluster with the closest centroid. The cluster centroids are then recomputed as the mean of the faces in the cluster. The algorithm continues to iterate until the cluster assignments no longer change, resulting in a final set of clusters. We can see the figure 3.2 in which 'person0' and 'person1' folders were created, and in these two folders, we stored the faces of two different persons that were detected through this algorithm. This approach allowed us to efficiently process large amounts of images that were extracted from the video data.

3.3.3 Head Pose Estimation

The determination of head position is a crucial component of our system. By determining the head position of the participants, we will be able to determine the direction of their gaze. If the person does not look at the screen for an extended period of time, we can assume that they are not paying sufficient attention to the lecture. For this module, we utilized the 6DRepNet model for unconstrained Head Pose estimation. Here, the 6DRepNet model is developed using Pytorch. The backbone of this model, 6DRepNet is RepVGG[21].

RepVGG

RepVGG is a convolutional neural network architecture having a VGG-like inference time body and is composed of a stack of 3*3 convolution and ReLU. The training time of this model has a topology which is multi-branched. A structural re-parameterization is used to realize the decoupling of the inference-time and the training-time and thus the model is named RepVGG. RepVGG is a simple ConvNet, while simple ConvNets have lower accuracy than the more complicated ones, they also have some significant advantages. Simple ConvNets such as RepVGG are faster as it does not introduce extra overheads such as synchronization and kernel launching and uses only one fragmented operator. They are also memory economical as it allows the occupied memory used by the inputs to be immediately released as soon as the operation is finished. Lastly simple ConvNets are flexible as it allows free configuration of every convolutional layer according to the requirements and allows pruning to achieve better efficiency-performance trade-off.

RepVGG has a plain topology like VGG without containing any branches as each layer only uses the preceding layer's output as input and passes its own output to the next layer.The body of RepVGG uses only ReLU and 3*3 convolution.In the concrete architecture of RepVGG,no compound scaling, manual refinement,automatic search, or other heavy designs are used to instantiate the concrete architecture.

Stage	Output size	RepVGG-A	RepVGG-B
1	112 x 112	1 x min(64, 64a)	1 x min(64, 64a)
2	56 x 56	2 x 64a	4 x 64a
3	28 x 28	4 x 128a	6 x 128a
4	14 x 14	14 x 256a	16 x 256a
5	7 x 7	1 x 512b	1 x 512b

Table 3.1: Architectural specification of RepVGG. $2 \times 64a$ means stage2 contains 2 layers each having 64a channels.

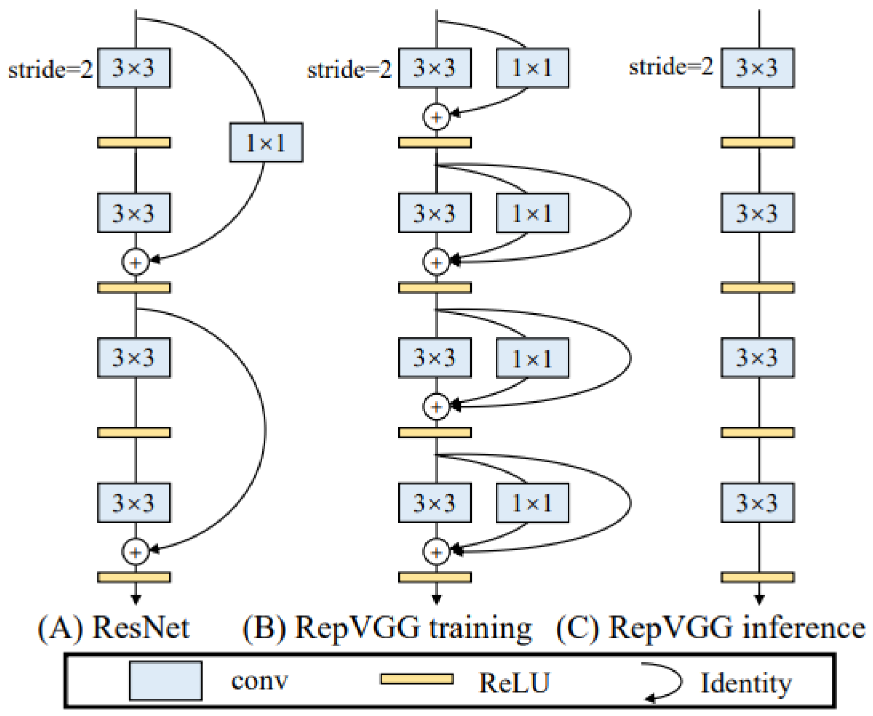


Figure 3.4: RepVGG architecture. It consists of 5 stages and down-sampling is conducted by stride-2 convolution when each stage begins.

RepVGG improves its efficiency by the use of structural re-parameterization. Structural re-parameterization means conversion of the architecture from one form to another form through transformation of the parameters. The training time of RepVGG is constructed using 1*1 branches and identities, structural-parameterization allows the branches to be removed. Transformation is done after training by the use of simple algebra. An identity branch can be perceived as a degraded 1*1 convolution and furthermore the identity branch that comes after can be also regarded as a degraded 3*3 convolution. Using this a single 3*3 kernel can be constructed which contains the trained parameters of the 3*3 original kernel, 1*1 and identity branches and batch normalization layers. Hence the model which is transformed contains a stack of 3*3 convolution layers which may be used for deployment and testing.

6DRepNet

6DRepNet is a cutting-edge deep learning model for 6D pose estimation of objects in an RGB-D image. It is designed to handle the challenges of estimating the 6D pose of objects in cluttered environments, where traditional methods may fail. One of the key innovations of 6DRepNet is its use of a rotation matrix representation to predict the precise head orientations of an object without the need for landmarks. This allows for full pose regression with a nine-parameter matrix without encountering any ambiguity issues.

The paper[23] presents a new method for head pose estimation that is based on a deep learning model called 6DRepNet. The model is trained using a new 6D rotation representation, which encodes the 3D rotation of the head as a vector of six parameters. The new representation is more robust to large variations in head pose and allows the model to estimate the 6D head pose (3D rotation and 3D translation) of a person from a single 2D image. The 6DRepNet model is trained on a large dataset of images of people with varying head poses and is then tested on several benchmark datasets. The results show that the 6DRepNet model is able to achieve state-of-the-art performance in estimating the 6D head pose of a person from a single 2D image. The authors also present an analysis of the different components of the model and show that the new 6D rotation representation is a key factor in the model's improved performance.

6D in 6DRepNet refers to the six degrees of freedom that are used to describe the pose of an object in 3D space. These six degrees of freedom include three for translation (x, y, z) and three for rotation (roll, pitch, yaw). In other words, 6D pose estimation is the process of determining the position and orientation of an object in 3D space, which is represented by its 3D translation and 3D rotation.

The use of a rotation matrix representation also simplifies the network architecture, making it more robust and efficient. This is because it eliminates the need for performance stabilizing techniques, such as the discretization of rotation variables into a classification problem. This also allows for easy conversion of network issues to other rotation-related tasks.

To further improve the accuracy of the model, 6DRepNet employs a novel repulsion loss function that is based on the geodesic distance on the SO(3) manifold. This loss function penalizes the network in relation to the geometry of the SO(3) manifold, which results in more accurate pose estimations. Additionally, the model enforces the orthogonality constraint $RR^T = I$ on the rotation matrix representation, which helps to ensure that the predicted rotation matrix is valid.

Another important aspect of 6DRepNet is its use of a compressed 6D representation, which is obtained by dropping the final column vector of the rotation matrix. This approach has been shown to result in smaller errors for direct regression, and also simplifies the network architecture. The compressed 6D representation is then transformed into a rotation matrix using the Gram-Schmidt mapping. This enables the model to effectively handle the challenges of large object datasets.

Furthermore, 6DRepNet also employs a data augmentation strategy to generate synthetic training data, further enhancing its performance. This enables the model to generalize well and improve its robustness in real-world scenarios.

Overall, 6DRepNet is a powerful and efficient model that uses state-of-the-art techniques to achieve high accuracy and robustness in cluttered environments. It is named 6DRepNet as it is influenced by the 6D representation method, and it's a powerful tool for robotic grasping and manipulation tasks in cluttered environments. Below is a model that compares 6DRepNet with other models [23].

3.3.4 Eye Gaze Estimation

Eye gaze tracking is one of the main components of our system. By tracking the eye gaze, we will get to know if the person is looking in the direction of the screen or not. If the participant is not looking in the direction of the screen for a relatively

		AFLW2000 dataset					BIWI dataset					BIWI 70/30 dataset			
Models	FULL RANGE	Yaw	Pitch	Roll	MAE		Yaw	Pitch	Roll	MAE		Yaw	Pitch	Roll	MAE
HopeNet(=2)	N	6.47	6.56	5.44	6.16		5.17	6.98	3.39	5.18		-	-	-	-
HopeNet(=1)	N	6.92	6.64	5.67	6.41		4.81	6.61	3.27	4.9		3.29	3.39	3	3.23
FSA-Net	N	4.5	6.08	4.64	5.07		4.27	4.96	2.76	4		2.89	4.29	3.6	3.6
HPE	N	4.8	6.18	4.87	5.28		3.12	5.18	4.57	4.29		-	-	-	-
QuatNet	N	3.97	5.62	3.92	4.5		2.94	5.49	4.01	4.15		-	-	-	-
WHENet-V	N	4.44	5.75	4.31	4.83		3.6	4.1	2.73	3.48		-	-	-	-
WHENet	Y/N	5.11	6.24	4.92	5.42		3.99	4.39	3.06	3.81		-	-	-	-
TriNet	Y	4.04	5.77	4.2	4.67		4.11	4.76	3.05	3.97		2.93	3.04	2.44	2.8
FDN	N	3.78	5.61	3.88	4.42		4.52	4.42	2.56	3.93		3	3.98	2.88	3.29
6DRepNet	Y	3.63	4.91	3.37	3.97		3.24	3.97	2.68	3.47		2.69	2.92	2.36	2.66

Table 3.2: Table comparing the 6DRepNet model with other models. The models were trained on the 300W-LP dataset and tested on the AFLW2000,BIWI and BIWI 70/30 dataset

long time then it is quite obvious that the person is not paying attention to what is being shown on the screen. So eye gaze tracking will be able to give us important information through which we will be able to determine the engagement level of the participants. For our research, we have used the ‘MediaPipe’ framework for eye gaze tracking.

Developed by Google, MediaPipe is an open-source framework that is cross-platform and designed for multimodal (i.e. pictures, video, and audio) applied machine learning pipelines. It comprises a collection of pre-built, reusable components for common machine-learning tasks such as object detection, facial landmark detection, and hand tracking. Furthermore, it enables developers to build, deploy, and run machine learning models on various media types such as live video streams, images, and audio. In addition, MediaPipe offers a user-friendly interface that makes it easy to create and deploy unique machine-learning models.

For eye gaze detection MediaPipe at first, uses computer vision algorithms to locate the subject’s eyes and face [14]. Typically, a combination of methods such as haar cascades, deep learning-based object detection, and facial landmark detection is used for this. MediaPipe offers a solution called FaceMesh that estimates 468 3D face landmarks in real time. It uses machine learning (ML) to infer the 3D facial surface and only needs one camera input—a specialized depth sensor is not required. The method provides the real-time speed necessary for live experiences by combining GPU acceleration across the pipeline with lightweight model architectures. After a mesh of approximate face geometry is generated, the eye region is isolated for use in the iris tracking step.

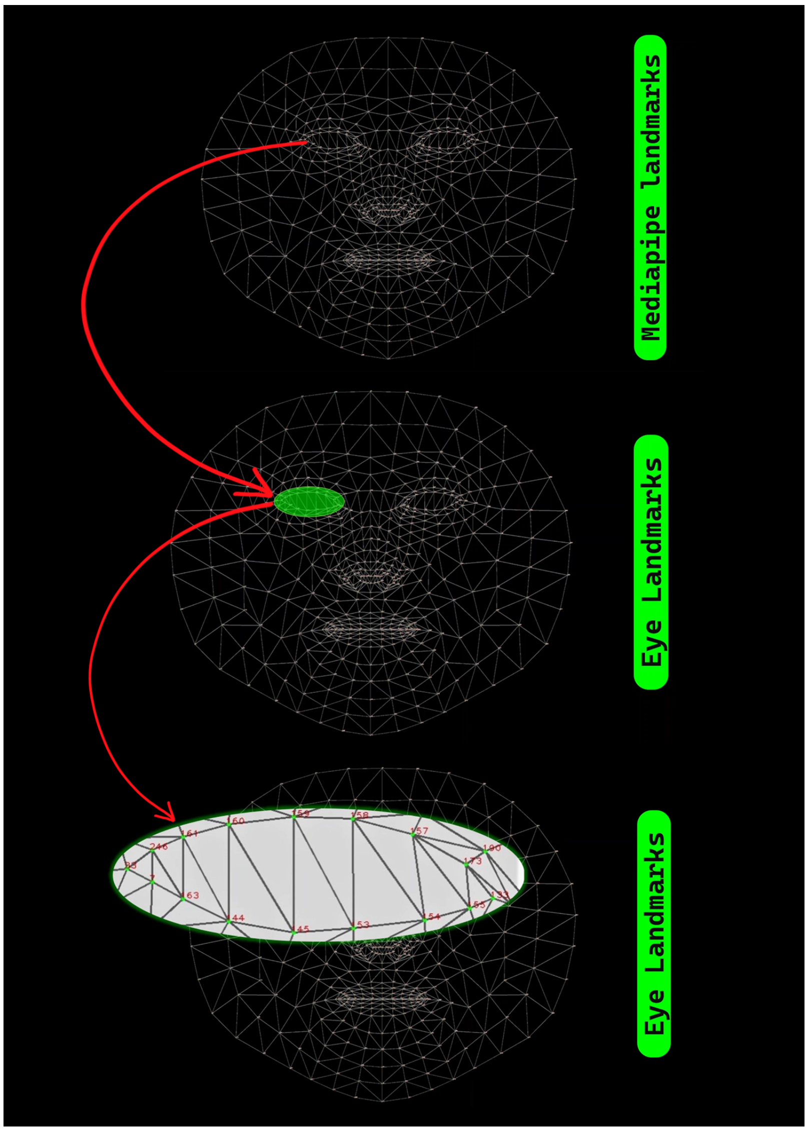


Figure 3.5: MediaPipe Landmarks of face and eye

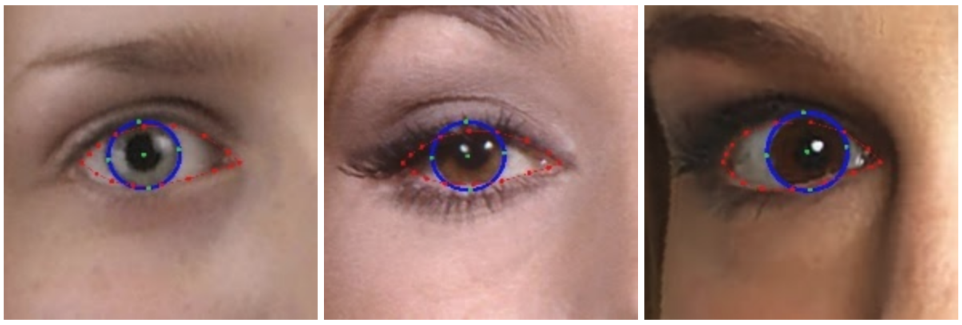


Figure 3.6: Eye region annotated with eyelid (red) and iris (blue) contours

For iris tracking the pipeline is constructed as a MediaPipe graph that makes use of a dedicated iris-and-depth renderer subgraph, an iris landmark subgraph from the iris landmark module, and a face landmark subgraph from the face landmark module [19]. An internal face detection subgraph from the face detection module is used by the face landmark subgraph. Through this process a set of 478 3D landmarks is generated, 468 of these are from the MediaPipe face mesh. The landmarks around the eyes are further polished and 10 additional iris landmarks are added at the end with 5 for each eye. Without additional hardware, MediaPipe Iris can accurately calculate the metric distance of a subject from the camera with less than a 10% error.

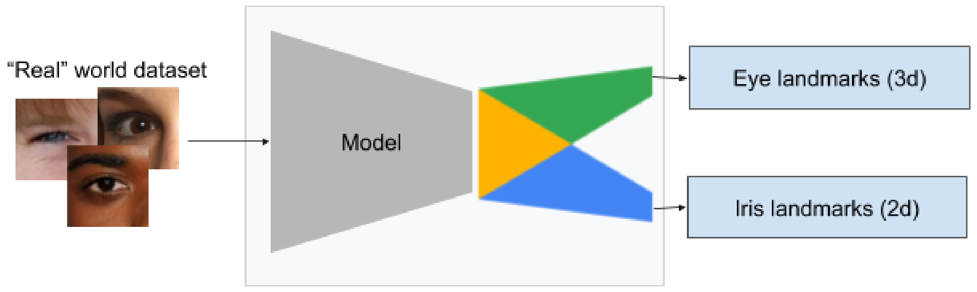


Figure 3.7: Cropped eye regions form the input to the model, which predicts landmarks via separate components

Chapter 4

Implementation

The challenge we were given was to find a way to measure how engaged a teacher and student were in a Google Meet and Zoom session. The term 'engagement' refers to both the students and the instructor's degree of involvement in the learning process. It is a very abstract task. In this study we tried to determine how intently both the student and the instructor were engaged. For this case, we used the MediaPipe framework's eye gaze, the 6DRepNet pre-trained model, and Screen sharing detection.

Screen sharing detection can give an indication of how much the teacher is sharing the screen during the online tutoring session, and how much the student is interacting with the shared content. It can be done by analyzing the screen recording of the tutoring session by using contours.

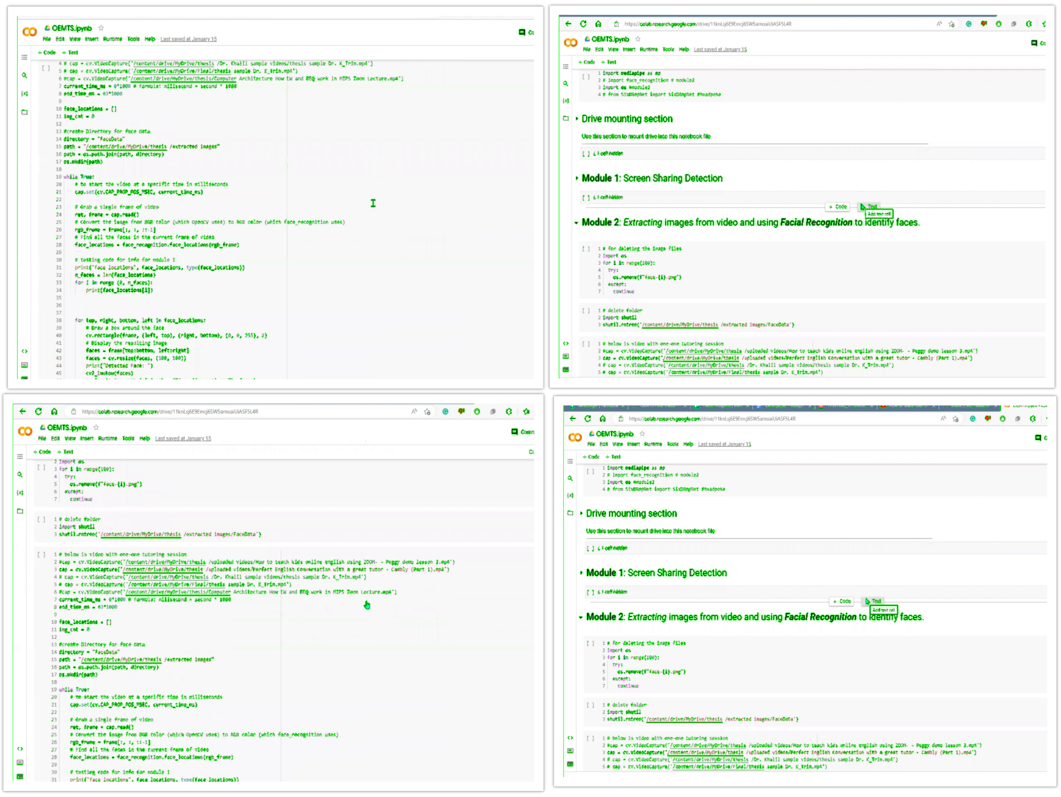
The MediaPipe framework uses computer vision techniques to track the student's eye gaze from video streams. It can detect the location of the eyes and the pupils and can estimate the gaze direction. This information can be used to determine if the student is looking at the teacher, the screen, or somewhere else.

The 6DRepNet pre-trained model, on the other hand, uses machine learning algorithms to estimate the head position from a video frame. The model is trained on a large dataset of head images and can estimate the yaw, pitch, and roll angles of the head. These angles give an indication of the head rotation and can be used to determine if the student is actually looking at the screen or not.

Due to the requirements of our goal, we had to fulfill multiple purposes in our system. We have separated those purposes into 5 different modules to properly divide the workforce and to organize the system. These modules are: 1) Screen share detection, 2) Face recognition, 3) Head Pose and eye-gaze estimation, and 4) Attention estimation and report generation.

4.1 Screen Share Detection

For Screen sharing, we used the 'Contour' calculation. Contours are the boundaries of an object in an image and can be used to detect the edges of a shared screen. We have set a threshold to calculate the total contours from our input video to determine whether the screen is being shared or not.


OEMTS.ipynb
File Edit View Insert Runtime Tools Help Last saved at January 15
+ Code + Text

```
# cap = cv.VideoCapture('/content/drive/MyDrive/thesis /Dr. Khalil sample videos/thesis sample Dr. K_Trim.mp4')
# cap = cv.VideoCapture('/content/drive/MyDrive/Final/thesis sample Dr. K_Trim.mp4')
#cap = cv.VideoCapture('/content/drive/MyDrive/thesis/Computer Architecture How LW and BEQ work in MIPS Zoom Lecture.mp4')
current_time_ms = 0*1000 # formula: milisecond = second * 1000
end_time_ms = 63*1000

face_locations = []
img_cnt = 0

#create Directory for face data
directory = "FaceData"
path = "/content/drive/MyDrive/thesis /extracted images"
path = os.path.join(path, directory)
os.mkdir(path)

while True:
    # to start the video at a specific time in milliseconds
    cap.set(cv.CAP_PROP_POS_MSEC, current_time_ms)

    # Grab a single frame of video
    ret, frame = cap.read()
    # Convert the image from BGR color (which OpenCV uses) to RGB color (which face_recognition uses)
    rgb_frame = frame[:, :, ::-1]
    # Find all the faces in the current frame of video
    face_locations = face_recognition.face_locations(rgb_frame)

    # testing code for info for module 1
    print("face locations", face_locations, type(face_locations))
    n_faces = len(face_locations)
    for i in range (0, n_faces):
        print(face_locations[i])



    for top, right, bottom, left in face_locations:
        # Draw a box around the face
        cv.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)
        # Display the resulting image
        faces = frame[top:bottom, left:right]
        faces = cv.resize(faces, (100, 100))
        print("Detected Face: ")
        cv2_imshow(faces)
```

https://colab.research.google.com/drive/11knLsj6E9Emcj6SW5arnoaiUiASF5L4R
OEMTS.ipynb
File Edit View Insert Runtime Tools Help Last saved at January 15
+ Code + Text

```
import mediapipe as mp
# import face_recognition # module2
import os #module2
# from SixDRepNet import SixDRepNet #headpose
```

Drive mounting section

Use this section to mount drive into this notebook file

↳ 1 cell hidden

Module 1: Screen Sharing Detection

↳ 1 cell hidden

Module 2: *Extracting* images from video and using ***Facial Recognition*** to identify faces.

```
# for deleting the image files
import os
for i in range(100):
  try:
    os.remove(f"face-{i}.png")
  except:
    continue
```

```
# delete folder
import shutil
shutil.rmtree('/content/drive/MyDrive/thesis /extracted images/FaceData')
```

```
# below is video with one-one tutoring session
#cap = cv.VideoCapture('/content/drive/MyDrive/thesis /uploaded videos/How to teach kids online english using ZOOM- - Peggy demo lesson 3.mp4')
cap = cv.VideoCapture('/content/drive/MyDrive/thesis /uploaded videos/Perfect English Conversation with a great tutor - Cambly (Part 1).mp4')
# cap = cv.VideoCapture('/content/drive/MyDrive/thesis /Dr. Khalil sample videos/thesis sample Dr. K_Trim.mp4')
# cap = cv.VideoCapture('/content/drive/MyDrive/Final/thesis sample Dr. K_Trim.mp4')
```

https://colab.research.google.com/drive/11knLsj6E9Emcj6SW5arnoaiUiASF5L4R
OEMTS.ipynb
File Edit View Insert Runtime Tools Help Last saved at January 15
+ Code + Text

```
import os
for i in range(100):
  try:
    os.remove(f"face-{i}.png")
  except:
    continue
```

```
# delete folder
import shutil
shutil.rmtree('/content/drive/MyDrive/thesis /extracted images/FaceData')
```

```
# below is video with one-one tutoring session
#cap = cv.VideoCapture('/content/drive/MyDrive/thesis /uploaded videos/How to teach kids online english using ZOOM- - Peggy demo lesson 3.mp4')
cap = cv.VideoCapture('/content/drive/MyDrive/thesis /uploaded videos/Perfect English Conversation with a great tutor - Cambly (Part 1).mp4')
# cap = cv.VideoCapture('/content/drive/MyDrive/thesis /Dr. Khalil sample videos/thesis sample Dr. K_Trim.mp4')
# cap = cv.VideoCapture('/content/drive/MyDrive/Final/thesis sample Dr. K_Trim.mp4')
#cap = cv.VideoCapture('/content/drive/MyDrive/thesis/Computer Architecture How LW and BEQ work in MIPS Zoom Lecture.mp4')
current_time_ms = 0*1000 # formula: milisecond = second * 1000
end_time_ms = 63*1000

face_locations = []
img_cnt = 0

#create Directory for face data
directory = "FaceData"
path = "/content/drive/MyDrive/thesis /extracted images"
path = os.path.join(path, directory)
os.mkdir(path)

while True:
    # to start the video at a specific time in milliseconds
    cap.set(cv.CAP_PROP_POS_MSEC, current_time_ms)

    # Grab a single frame of video
    ret, frame = cap.read()
    # Convert the image from BGR color (which OpenCV uses) to RGB color (which face_recognition uses)
    rgb_frame = frame[:, :, ::-1]
    # Find all the faces in the current frame of video
    face_locations = face_recognition.face_locations(rgb_frame)

    # testing code for info for module 1
    print("face locations", face_locations, type(face_locations))
```

https://colab.research.google.com/drive/11knLsj6E9Emcj6SW5arnoaiUiASF5L4R
OEMTS.ipynb
File Edit View Insert Runtime Tools Help Last saved at January 15
+ Code + Text

```
import mediapipe as mp
# import face_recognition # module2
import os #module2
# from SixDRepNet import SixDRepNet #headpose
```

Drive mounting section

Use this section to mount drive into this notebook file

↳ 1 cell hidden

Module 1: Screen Sharing Detection

↳ 1 cell hidden

Module 2: *Extracting* images from video and using ***Facial Recognition*** to identify faces.

```
# for deleting the image files
import os
for i in range(100):
  try:
    os.remove(f"face-{i}.png")
  except:
    continue
```

```
# delete folder
import shutil
shutil.rmtree('/content/drive/MyDrive/thesis /extracted images/FaceData')
```

```
# below is video with one-one tutoring session
#cap = cv.VideoCapture('/content/drive/MyDrive/thesis /uploaded videos/How to teach kids online english using ZOOM- - Peggy demo lesson 3.mp4')
cap = cv.VideoCapture('/content/drive/MyDrive/thesis /uploaded videos/Perfect English Conversation with a great tutor - Cambly (Part 1).mp4')
# cap = cv.VideoCapture('/content/drive/MyDrive/thesis /Dr. Khalil sample videos/thesis sample Dr. K_Trim.mp4')
# cap = cv.VideoCapture('/content/drive/MyDrive/Final/thesis sample Dr. K_Trim.mp4')
```

Figure 4.1: Screen Extraction from Video Frames

4.2 Face Recognition

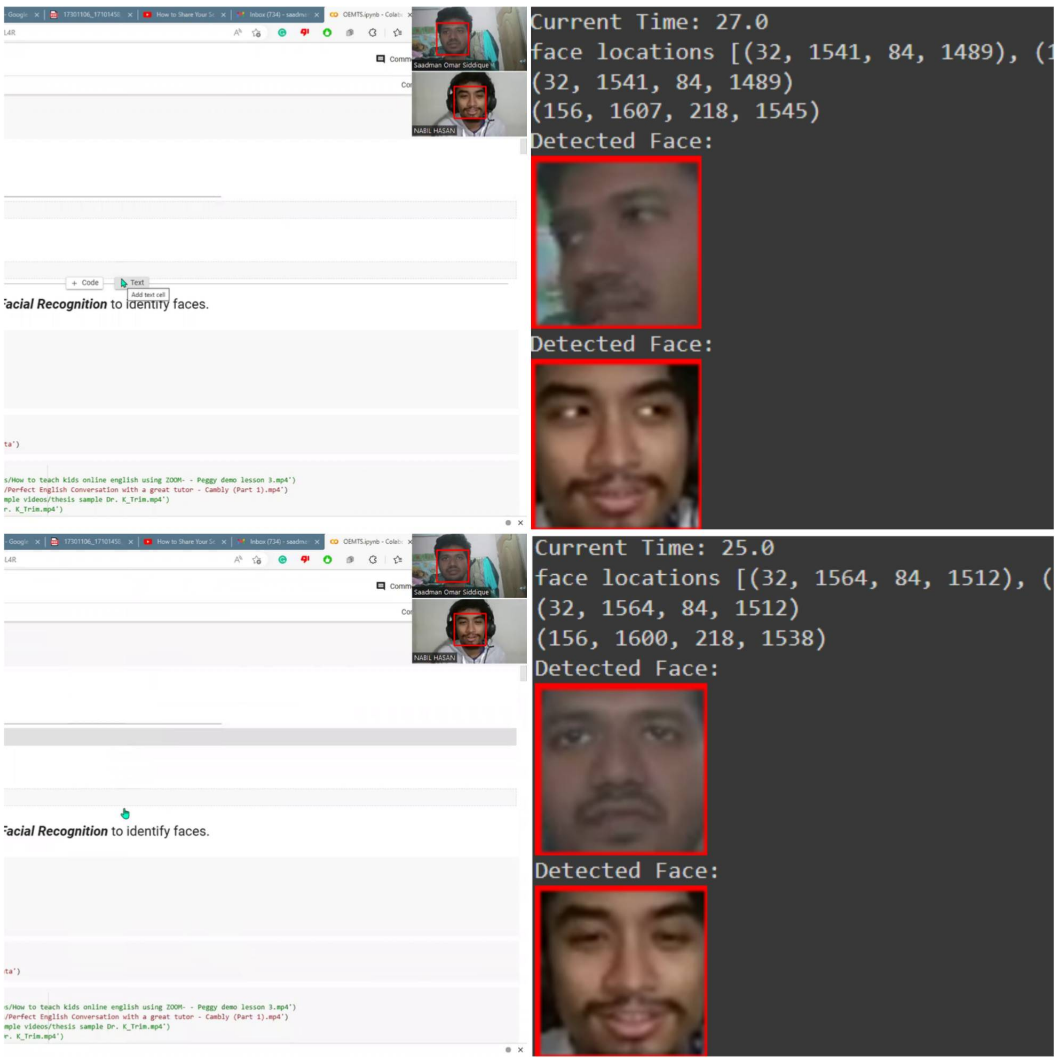
We used the face recognition python library as an effective tool for detecting and identifying faces in video and images which uses sophisticated algorithms to recognize and extract facial features from images. Our approach also employed deep learning algorithms to identify faces and match them against a database of known faces. We leveraged the power of this library to detect and extract faces from the frames of the video input. Our detected faces were processed and analyzed through the use of the KMeans library for clustering. This allowed us to group the detected faces according to their similarities and categorize them into subfolders for further analysis.

4.3 Head Position and Eye Gaze estimation

The implementation of Head Position and Eye Gaze estimation is discussed.

4.3.1 Head Position Estimation

On our very first we attempted to use the VGG-16 architecture, a convolutional neural network (CNN) that was trained for image classification tasks, to estimate the head position in our online tutoring sessions. The VGG-16 architecture uses a series of convolutional layers, max-pooling layers, and fully connected layers to extract features from an image and classify it into one of several predefined categories.



Facial Recognition to identify faces.

```
ta')
s/How to teach kids online english using ZOOM- - Peggy demo lesson 3.mp4')
/Perfect English Conversation with a great tutor - Cambly (Part 1).mp4')
mple videos/thesis sample Dr. K_Trim.mp4')
r. K_Trim.mp4')
```

```
Current Time: 27.0
face locations [(32, 1541, 84, 1489), (1
(32, 1541, 84, 1489)
(156, 1607, 218, 1545)
Detected Face:
Detected Face:
```

Facial Recognition to identify faces.

```
ta')
s/How to teach kids online english using ZOOM- - Peggy demo lesson 3.mp4')
/Perfect English Conversation with a great tutor - Cambly (Part 1).mp4')
mple videos/thesis sample Dr. K_Trim.mp4')
r. K_Trim.mp4')
```

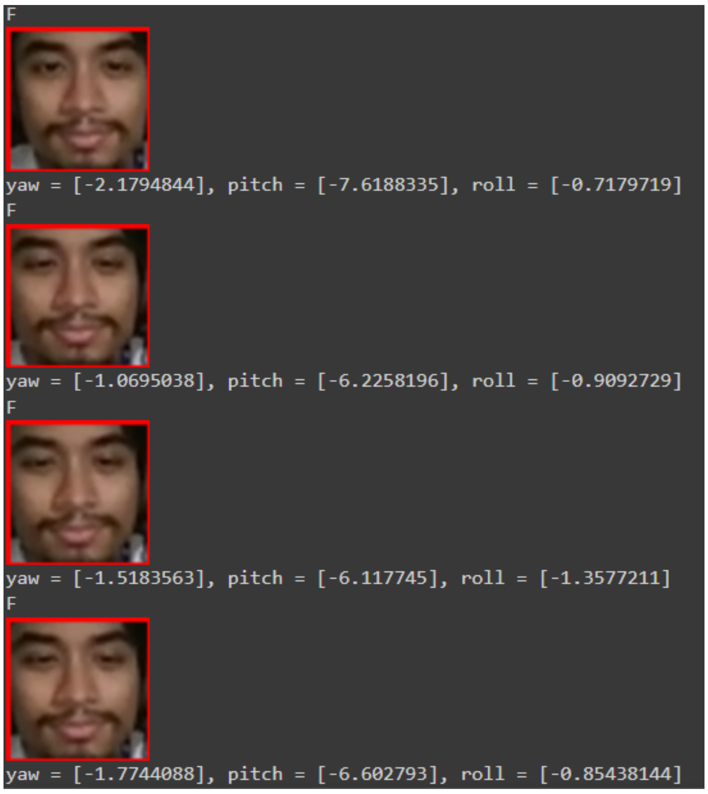
```
Current Time: 25.0
face locations [(32, 1564, 84, 1512), (
(32, 1564, 84, 1512)
(156, 1600, 218, 1538)
Detected Face:
Detected Face:
```

Figure 4.2: Face Extraction From Video Frames

However, our attempts to use VGG-16 for head position estimation were not successful as the accuracy was not sufficient. One of the reasons for this is that VGG-16 was not designed or trained specifically for the task of head position estimation, but rather for object classification using the ImageNet dataset which contains over 14 million images and 1,000 object categories. Therefore, it may not have been able to extract the necessary features for head position estimation or handle the variations in head position. Additionally, the ImageNet dataset does not have enough head position data for VGG-16 to learn from.

Despite this, we continued to explore other methods and eventually found a more suitable approach, the facial landmark detection component of MediaPipe, which showed better results in terms of accuracy and robustness. This method, which will be discussed in detail in the following sections, proved to be more effective for our task of head position estimation in online tutoring sessions.

During the course of this research, one of our primary objectives was to devise a strategy for reliably assessing the head position of participants in online tutoring sessions. In order to accomplish this, we made use of the facial landmark detection



```
F
yaw = [-2.1794844], pitch = [-7.6188335], roll = [-0.7179719]
F
yaw = [-1.0695038], pitch = [-6.2258196], roll = [-0.9092729]
F
yaw = [-1.5183563], pitch = [-6.117745], roll = [-1.3577211]
F
yaw = [-1.7744088], pitch = [-6.602793], roll = [-0.85438144]
```

Figure 4.3: Head Pose Detection from Video Frames

component of MediaPipe, which is a popular tool in the field of computer vision. This allowed us to recognize and extract pertinent facial landmarks from photos or video frames. This component identifies particular points on the face, such as the lips, eyes, and nose, by employing computer vision algorithms to do so.

After that, we used methods such as triangulation to estimate the head position by using the coordinates of the detected facial landmarks as known points. This allowed us to get a more accurate reading. The estimation of the location of the head was accomplished by using the distances that separated these places.

However, the accuracy of the head position estimation is dependent on a number of parameters including the quality of the image or video, the lighting conditions, and the performance of the facial landmark detection component. These issues are discussed more in the next paragraph. In spite of these difficulties, the method that was suggested displayed a relatively good accuracy in determining where the head was located. Despite this, the accuracy was not up to par because of the constraints imposed by the method as well as the difficulties encountered throughout the implementation.

It is important to point out that this method was the very first one that we attempted to utilize; later on, however, we switched to the 6DRepNet, which is more recent and has a higher degree of accuracy and fewer restrictions.

We employed a deep learning model called '6DRepNet' [23] in order to estimate the head position of a subject. The model was pre-trained and we fine-tuned it to improve its performance. The fine-tuning process involved training the model on specific datasets, namely the 300W-LP, AFLW2000, and BIWI datasets. These datasets contain a diverse range of head poses and facial features, which allows the model to accurately estimate the head position of a subject in a variety of conditions.

By using the '6DRepNet' with a fine-tuned model, we achieved a high level of precision and accuracy in our head position estimation. The model returns 3 values: yaw, pitch, and roll. By setting specific thresholds, we can predict the position of the head. For example, if the pitch value is greater than -10 and less than 10 and the yaw value is greater than -20 and less than 20, then the position of the head is considered to be forward. Similarly, if the yaw value is less than -20, then the position is considered to be left, and if the yaw value is greater than 20, then the position is considered to be right.

It is worth noting that these thresholds were determined based on our interaction and the nature of the task, and could be different for other use cases. It is important to validate the thresholds using a dataset and professional advice. This allowed us to get more accurate results on the head position whether it is in the right or left or forward or up or down position. We obtained our Head data from this Module.

4.3.2 Eye Estimation

Eye gaze estimation is a challenging task that involves determining the direction of a person's gaze based on their eye movements. For eye estimation, we looked through a combination of various models and research papers. However, it is important to note that despite these efforts, eye gaze tracking remains a complex task and current methods may not provide highly accurate results. There are various approaches to solving this problem, including using computer vision and machine learning techniques such as feature-based methods, model-based methods, and deep learning techniques.

Feature-based methods involve taking features from the eye area of an image and using them to estimate the direction of gaze. Methods like Haar cascades, optical flow, and deep learning can be used to do this.

Model-based methods are another approach that could be in use to figure out where someone is looking. To do this, we create a 3D model of the eye and head and use it to estimate the gaze direction. This can be done using techniques such as eye tracking or head pose estimation.

Deep learning techniques like convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have also been used to solve the gaze estimation problem. The model is trained on a set of images of eyes with their corresponding gaze directions.

We used Mediapipe for eye gaze estimation as it was easy to implement due to our limited time on our research. We were able to determine a person's gaze to some degree. We obtained the Eye data from this Module.

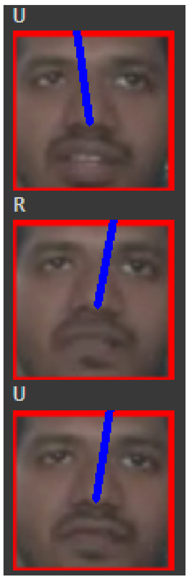


Figure 4.4: Head Pose estimation using MediaPipe and OpenCV

The figure above depicts the approach we used in our previous attempts, where it can be seen that the Head Position Estimatation implementation using MediaPipe and OpenCV was not very accurate.

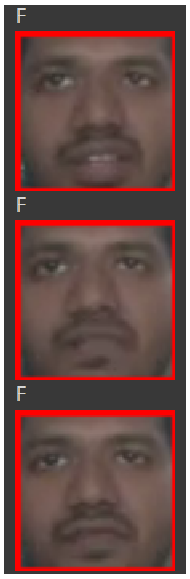


Figure 4.5: Head Pose estimation using 6DRepNet

The above figure refers to the method we used in our latest attempt, where we used a newer pre-trained model to determine the head position. We can see that the result is quite good.

4.4 Attention Estimation and Report Generation

Attention estimation is a process of determining the level of attention or focus of an individual in a given task or situation. We attempted to estimate the attention from the online tutoring session and to present the data visually.We used Head Position estimation, Eye Gaze estimation, and Screen Share detection to properly identify the Engagement between a tutor and a student.

With the results obtained from these methods, we presented the data in a visually-appealing format, such as a report, making it easy for supervisors to analyze and understand the level of attention of the students. This information can be used to identify instances of procrastination or distraction, allowing for early intervention and support. Furthermore, it allows for real-time monitoring of student's focus and engagement, providing a valuable tool for teachers and educators to optimize their teaching methods and improve students' learning outcomes.

Overall, our study highlights the importance of attention estimation in educational settings and provides a valuable tool for teachers and supervisors to monitor and enhance student engagement and focus during online tutoring session.

4.4.1 Attention Estimation

In our study, we wanted to find out how much students pay attention during an online tutoring session. To do this, we used the power of advanced computer vision techniques to track how the students moved their heads and where they looked. The "head data" from our Head estimation Module and the "eye data" from our Eye Gaze Estimation Module were then put through our algorithm, AttentionEstimator. Our algorithm does a good job of classifying the students' attention, which gives us valuable information about how engaged the students are and how they learn. This helps us improve how well the online tutoring system works. Figure 4.6 shows our Algorithm that we supervised.

Our algorithm examines blocks of attention data, with each block consisting of 3 consecutive attention values. It checks if all three values within a block are equal to zero. If this is the case, we assume that the individual is not paying attention during that specific time period, and we proceed to set all three values within that block to 1. This process is intended to clean and standardize the attention data, making it more consistent and easier to interpret. It is important to note that our algorithm heavily relies on accurate gaze estimation. The position of the eye on the screen plays a crucial role in determining the algorithm's performance. In this study, we focused on utilizing eye gaze data, which takes into account the position of the eye on the screen, to enhance the accuracy and effectiveness of our algorithm. For other eye gaze estimation our algorithm might need a slight adjustment.

4.4.2 Report Generation

The report generated from our study utilizes advanced data visualization techniques to effectively convey the results of the attention estimation and screen sharing detection methods. We used popular python libraries such as Mathplotlib and Pandas,

Figure 4.6 AttentionEstimator Algorithm

```
1: $att\_data1 \leftarrow []$
2: $att\_data2 \leftarrow []$
3: att_dict ← {'Fc': 1, 'Fl': 0, 'Fr': 0, 'Fcl': 0, 'Rc': 0, 'Rl': 1, 'Rr': 0, 'Rcl': 0,
   'Lc': 0, 'Ll': 0, 'Lr': 1, 'Lcl': 0, 'Uc': 1, 'Ul': 0, 'Ur': 0, 'Ucl': 0, 'Dc': 1, 'Dl':
   0, 'Dr': 0, 'Dcl': 0}
4: for $i$ in range(0, len(data1)) do
5:     $key \leftarrow data1[i] + eye\_data1[i]$
6:     $att\_data1$.append($att\_dict[key]$)
7: for $k \leftarrow 0, len(att\_data1)$ do
8:     if $att\_data1[k] \neq 0$ & $att\_data1[k+1] \neq 0$ & $att\_data1[k+2] \neq 0$ then
9:         $att\_data1[k] \leftarrow att\_data1[k+1] \leftarrow att\_data1[k+2] \leftarrow 1$
10: for $j$ in range(0, len(data2)) do
11:     $key \leftarrow data2[j] + eye\_data2[j]$
12:     $att\_data2$.append($att\_dict[key]$)
13: for $l \leftarrow 0, len(att\_data1)$ do
14:     if $att\_data2[l] \neq 0$ & $att\_data2[l+1] \neq 0$ & $att\_data2[l+2] \neq 0$ then
15:         $att\_data2[l] \leftarrow att\_data2[l+1] \leftarrow att\_data2[l+2] \leftarrow 1$
```

which are known for their powerful data visualization capabilities. The report includes beautiful visuals such as pie charts, which were used to represent the level of attentiveness during the online tutoring session. These charts clearly show the distribution of attention levels, making it easy for supervisors to identify patterns and trends in the data.

Additionally, we used line graphs to represent the results of the screen sharing detection. These graphs effectively display the duration of screen sharing during the session and allow for easy identification of instances of distraction or procrastination. The line graphs are interactive and can be zoomed in and out to see the specific time of screen sharing.

Overall, the report generated from our study is designed to be user-friendly and intuitive, making it easy for supervisors to understand and analyze the data. The report's visual representation of the data provides a clear and concise overview of the students' attention levels, allowing for effective monitoring and intervention.

Chapter 5

Results

The performance of our system was evaluated by processing four video files throughout the modules of our system and building the dataset. The results were then analyzed and the accuracy of the system was computed. The findings of this evaluation are presented below.

As discussed before, we have used four 45-second videos with the following features: Video 1- 2 persons with webcams and screen share on, Video 2- 1 person with webcam and screen share on, Video 3- 2 persons with webcams on, no screen share, Video 4- 2 persons without webcams on, screen share on at times.

Accuracy formula:

$$(Tested\, data/ground\, truth) * 100\%$$

Here, ground truth is the number of frames or data that has been manually checked for comparison against AI or algorithm data.

5.1 Accuracy calculation

5.1.1 Screen sharing detection

The videos are tested with our built algorithm using contour to detect the screen sharing of the recorded video.

	Screen sharing detection
Video 1	(44/45) * 100% = 97.78%
Video 2	(45/45) * 100% = 100%
Video 3	(37/45) * 100% = 82.22%
Video 4	(44/45) * 100% = 97.78%

Table 5.1: Accuracy calculation of the Screen Share detection module using Contour function

Possible causes of inaccuracy:
Video 1 - OpenCV missed one frame at the beginning.

Video 3 - Despite no screen share being present, our approach of counting the contours led to some frames being mistakenly detected as screen-sharing due to the high number of contours on the screen.
Video 4 - OpenCV again missed one frame at the beginning

5.1.2 Face detection

Running the videos through the face detection model, we tried to extract the face images. The detected faces are put in the formula against the real number of faces manually which is checked by us to calculate the accuracy.

	Face detection
Video 1	(66/88) * 100% = 75%
Video 2	(44/44) * 100% = 100%
Video 3	(81/88) * 100% = 92.05%
Video 4	(32/44) * 100% = 72.73%

Table 5.2: Accuracy calculation of the Face Detection module using Face_Recognition

Possible causes of inaccuracy:
Video 1 and 3 - Lower video quality, poor lighting conditions, or smaller resolution may have led to inaccuracies in detecting the faces in videos 1 and 3
Video 4 - Despite no face cams being present, the model mistakenly detected some irrelevant facial images in the shared screen as faces.

5.1.3 Face recognition and clustering

Using the Face_recognition model and K-means model, the detected and extracted faces from the previous module is here first recognized or identified and then clustered (organized) in separate folders based on their identification.

	Person	Face clustering estimation
Video 1	Person 1	(28/29) * 100% = 96.55%
	Person 2	(35/36) * 100% = 97.22%
Video 2	Person 1	(35/44) * 100% = 79.55%
Video 3	Person 1	(43/43) * 100% = 100%
	Person 2	(37/37) * 100% = 100%

Table 5.3: Accuracy calculation of face clustering/organizing based on person's face identity using K-means model

Possible causes of inaccuracy:
Video 1 - One image for each person was misplaced, probably due to similar facial features.
Video 2 - Probably due to the inaccuracy of the recognition model, some faces were not detected.

5.1.4 Head-position estimation

We employed the 6dRepNet model to estimate the head position of the dataset comprised of images of individuals extracted from videos in online tutoring sessions. The Head Position Estimation module was utilized to accurately estimate the head position for the analysis of the system's results.

	Person	Head-position estimation
Video 1	Person 1	(29/29) * 100% = 100%
	Person 2	(36/36) * 100% = 100%
Video 2	Person 1	(44/44) * 100% = 100%
Video 3	Person 1	(41/43) * 100% = 95.35%
	Person 2	(36/37) * 100% = 97.3%

Table 5.4: Accuracy calculation of the Head-position estimation using 6dRepNet

5.1.5 Eye-gaze estimation

Using google's MediaPipe, we estimated the eye-gaze of the persons' eyes.

	Person	Eye-gaze estimation
Video 1	Person 1	(5/29) * 100% = 17.24%
	Person 2	(24/36) * 100% = 66.67%
Video 2	Person 1	(36/44) * 100% = 81.82%
Video 3	Person 1	(24/43) * 100% = 55.81%
	Person 2	(29/37) * 100% = 78.38%

Table 5.5: Accuracy calculation of the eye-gaze estimation using mediapipe

Possible causes of lower accuracy:
It is possible that the accuracy of the system was impacted by the lower resolution of the recorded webcam footage, which resulted in a smaller dimension of the eye area. This smaller dimension could have made it more challenging for the system to accurately detect and analyze eye movements.

5.1.6 Overall Average Accuracy

Based on the accuracy formula, we estimated the average accuracy tested with the listed videos and expressed it in the following table. However, we did not use video 4 for result analysis in the case of Face recognition and Clustering, Head-position estimation, and Eye-gaze estimation since no faces were recognized due to missing camera footage.

Module	**Accuracy**
Screen Share detection	94.50%
Face detection	84.95%
Face recognition and Clustering	94.66%
Head-position estimation	98.53%
Eye-gaze estimation	59.98%
Overall system	86.52%

Table 5.6: Average accuracy of different modules

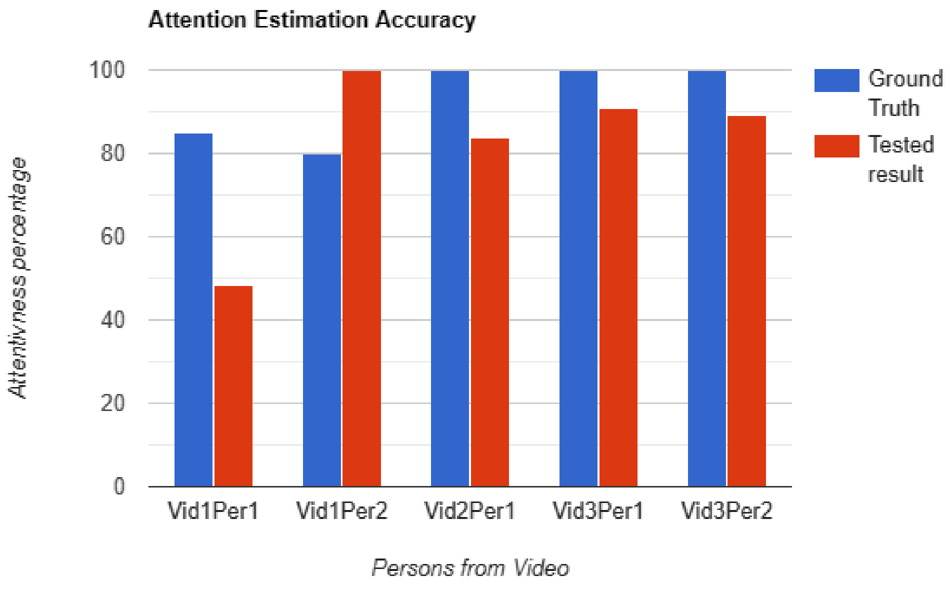


Figure 5.1: Attention Measurement Accuracy

The above line graph shows the contrast between the manual ground truth and the tested result provided by our system. Vid1Per1 and Vid1Per2 represent the two people in video 1, and similarly for the remaining videos. In the instance of Video 1, we can see that the system's accuracy was hampered by the lower quality of the recorded webcam footage, which resulted in a smaller size of the eye region. Furthermore, the performance of face recognition in video 1 was poor when compared to the other videos, resulting in lower overall accuracy in video 1 when compared to Video 2 and Video 3.

Video	Accuracy
1_Per1	56.82%
1_Per2	80%
2_Per1	83.7%
3_Per1	90.7%
3_Per2	89.2%
Overall Accuracy	80.08%

Table 5.7: Overall accuracy of the system

The above table shows the overall accuracy of the system run on 3 example videos. From this table, we can see that we achieved an overall moderate accuracy of 80.08%. This means that our algorithm detects the involvement level of both the student and instructor in a 1on1 online session with excellent accuracy. However, We recognize that there is always room for improvement. We believe that by fine-tuning the eye-gaze estimation model and improving the algorithm for attention assessment depending on the modules employed, we may obtain higher accuracy. We plan to further investigate these areas in future work to improve the overall performance of the system.

5.2 Screen share detection report

The pie charts show the total percentage of time the screen was shared in the target duration (around 45 seconds). For example, 95.6% screen shared means that during that 45-second duration, 95.6% of the time the screen was being shared. Pie charts and bar charts are demonstrated below for the tested videos.

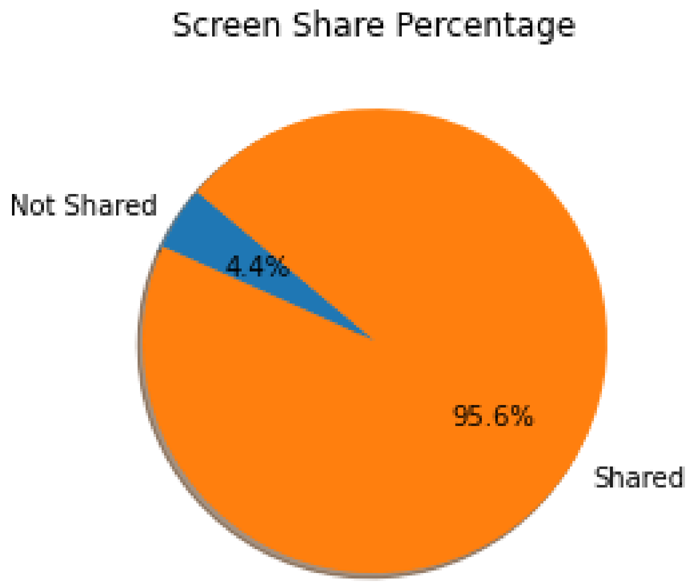


Figure 5.2: Screen Share percentage of Video 1

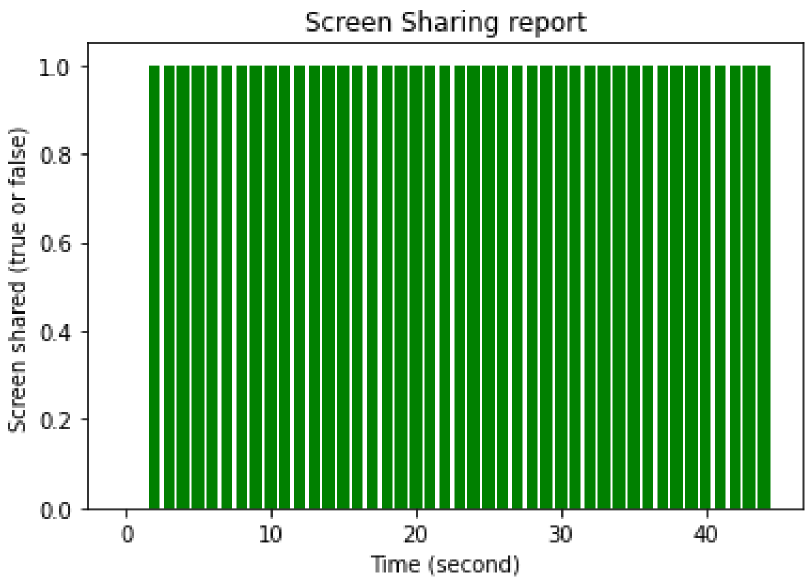


Figure 5.3: Screen Share timelapse of Video 1

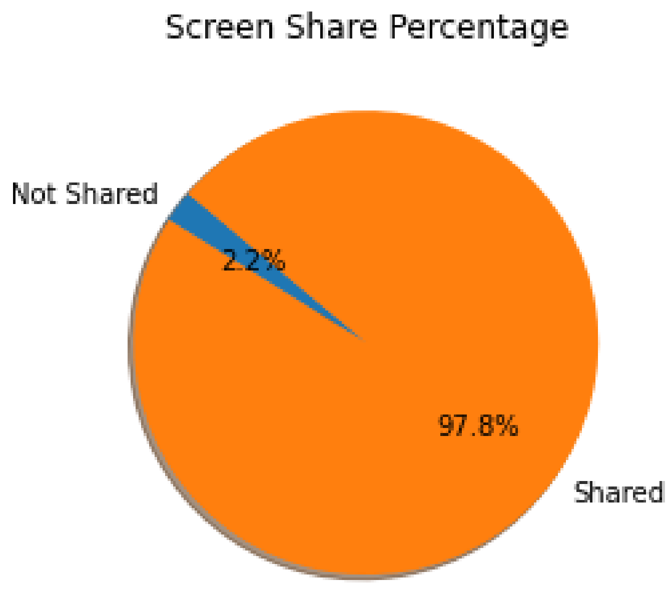


Figure 5.4: Screen Share percentage of Video 2

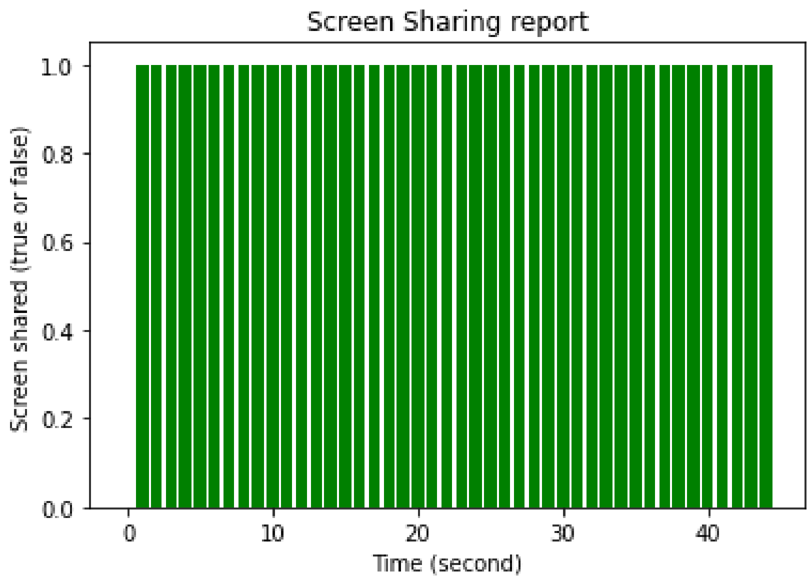


Figure 5.5: Screen Share timelapse of Video 2

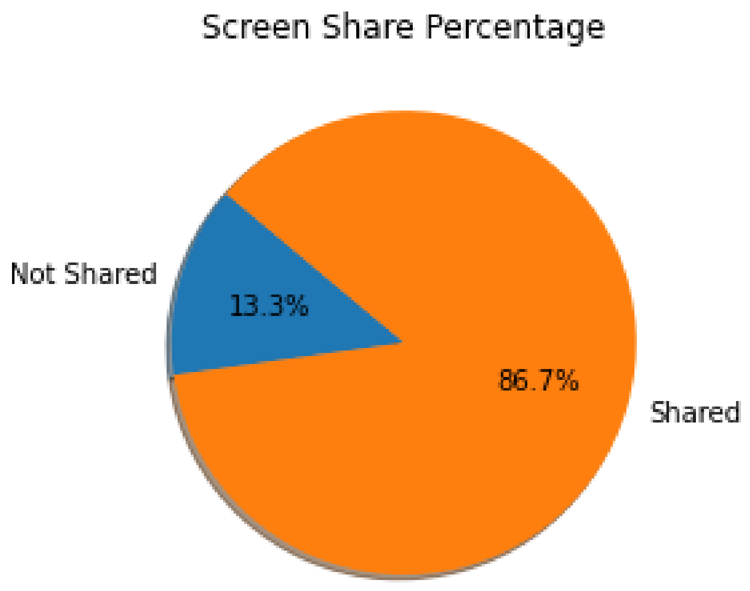


Figure 5.6: Screen Share percentage and Pie chart of Video 3

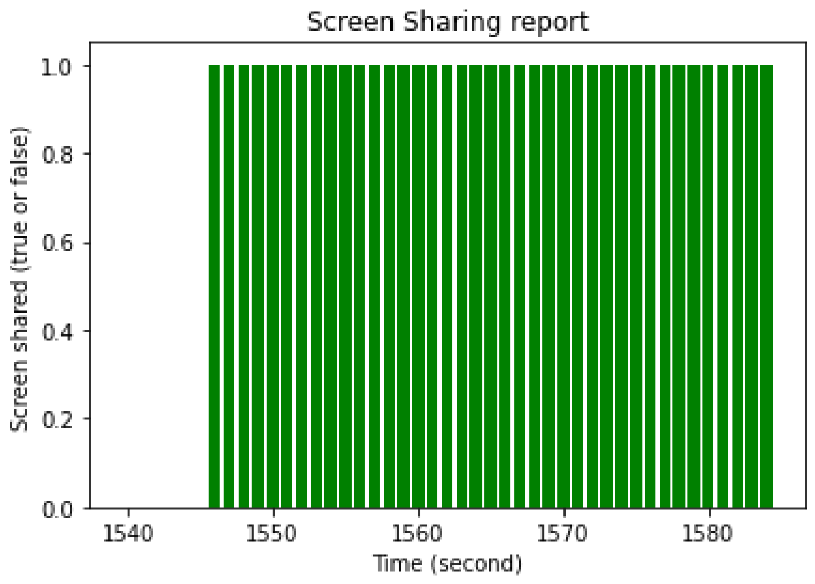


Figure 5.7: Screen Share percentage and bar chart of Video 3

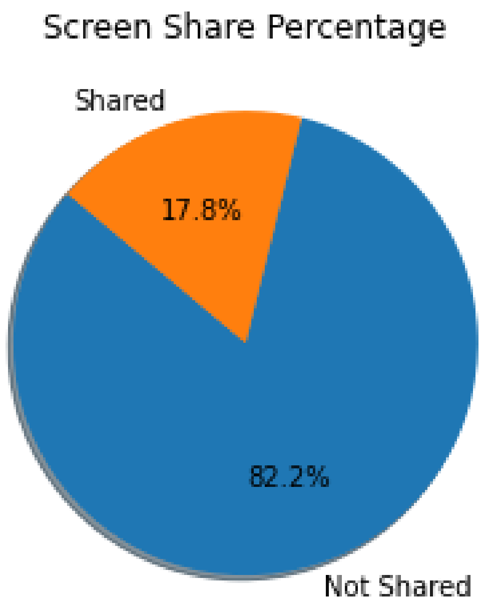


Figure 5.8: Screen Share percentage and Pie chart of Video 4

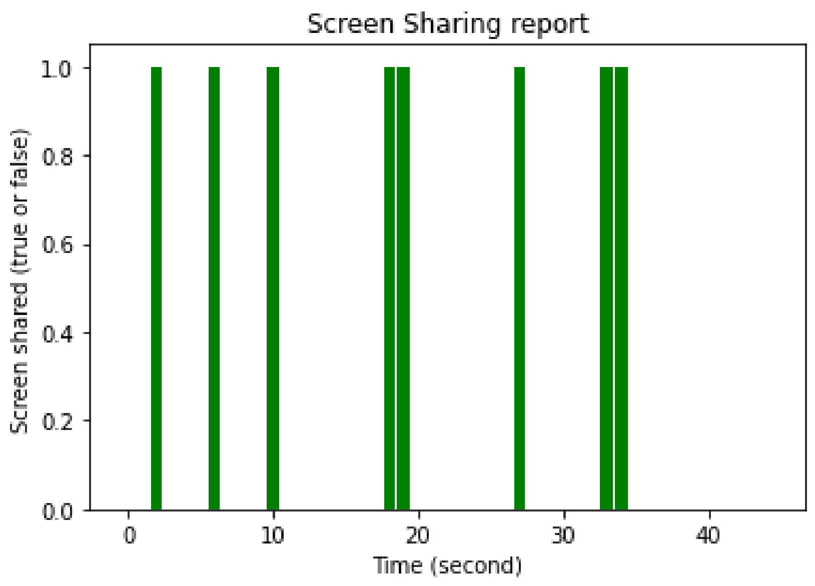


Figure 5.9: Screen Share percentage and bar chart of Video 4

Besides, the bar charts reveal the precise time periods during which screen share was recognized. For example, in video 4, we can see that the bar is green for a short period of time between 0 and 40 seconds. Those were the moments that Screen Share was detected, whereas it was not the rest of the time.
The above graphs are only the generated reports from our system, not the accuracy measures of the module. For accuracy measurements, we can refer to section 5.1.

5.3 Attentiveness report generation

For attention estimation, we first represented the raw data of attention, and then we applied various realistic restrictions specified in the implementation section to make the report more realistic and avoid anomalies. As a result, after imposing the attention limitations, the accuracy appears to have marginally increased in most situations. We have shown our calculated attention to the tested data in the pie charts of tested videos below.

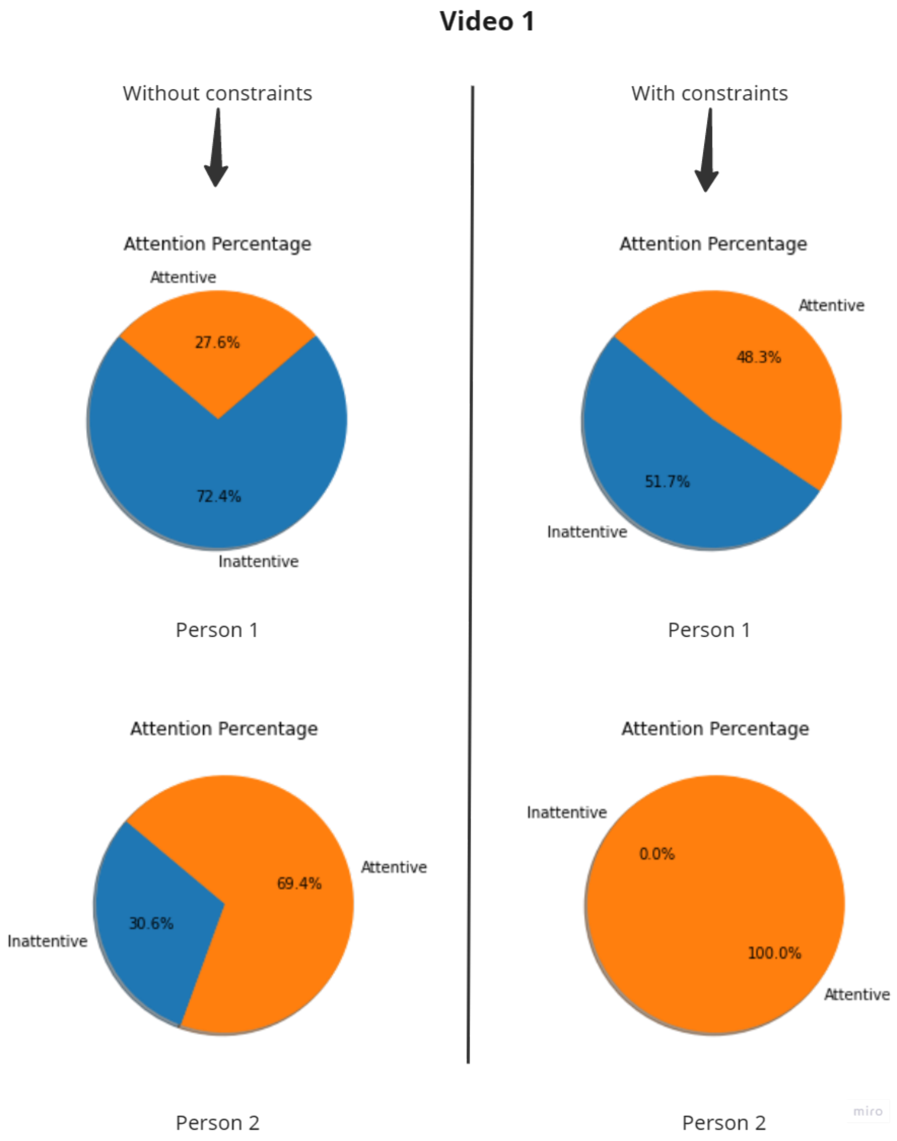


Figure 5.10: Attention percentage of Video 1 with and without constraints

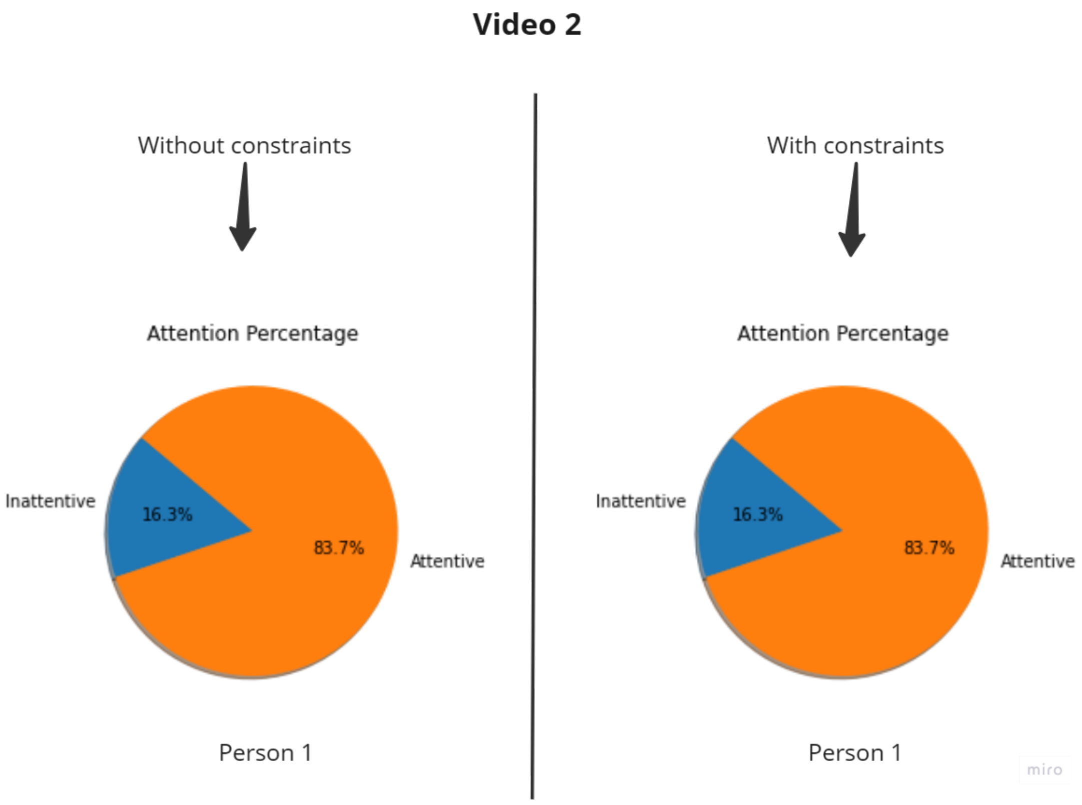


Figure 5.11: Attention percentage of Video 2 with and without constraints

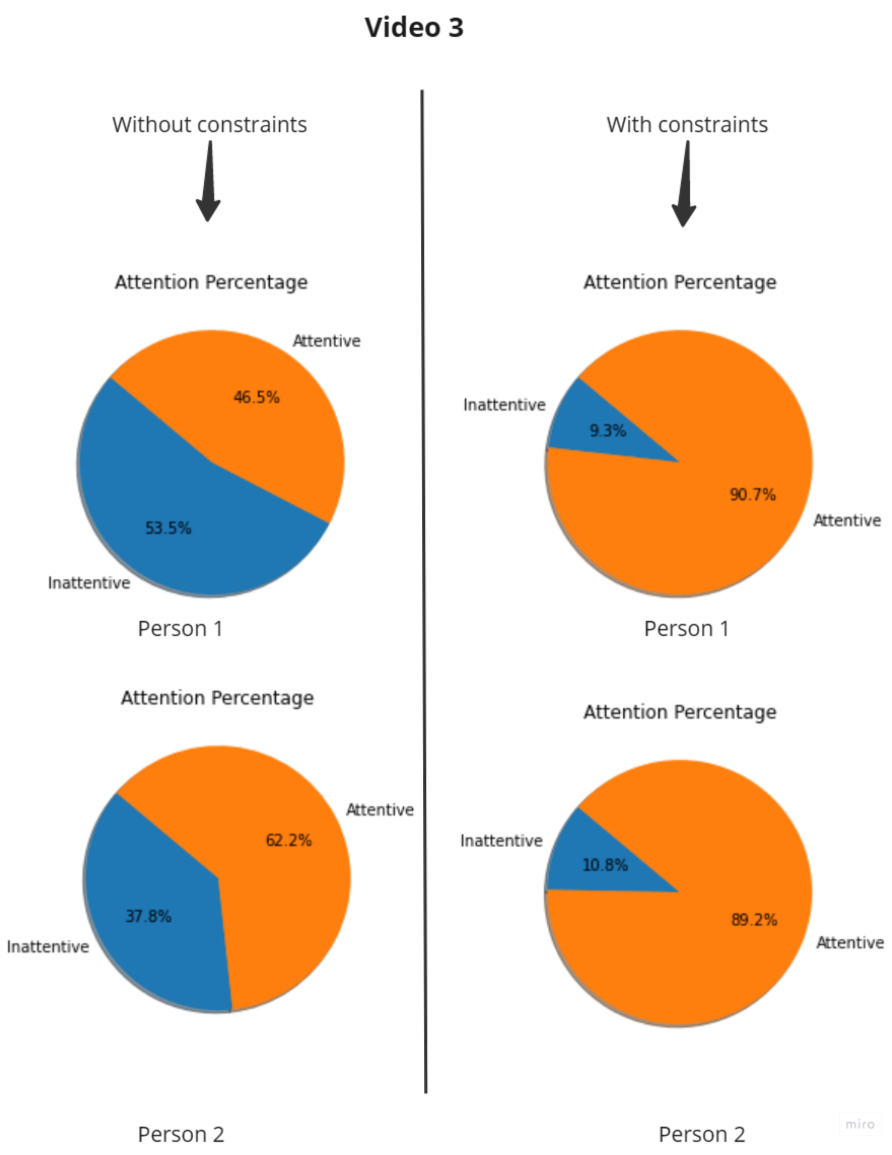


Figure 5.12: Attention percentage of Video 3 with and without constraints

After processing, we also created a graph output which is given below. Here, the black area represents the missing region. For instance, in screen share, the absence of screen sharing is shown by the dark area. Similarly, the black area in the instance of person 2 indicates that person 2 was not attentive at those times. This is done for an example video.

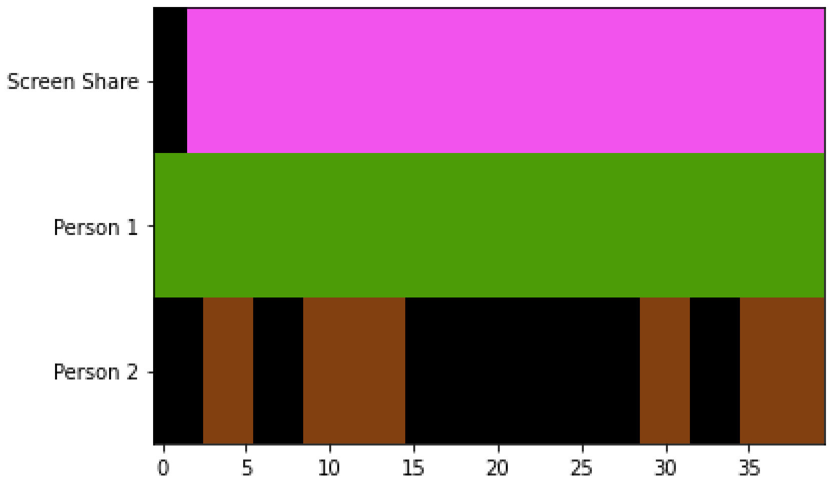


Figure 5.13: Screen Share percentage of Video 1 and Attention percentage

Chapter 6

Conclusion

In conclusion, the system we designed and researched for assessing the attention of students and tutors in online classes is a practical and effective method for measuring engagement during online tutoring sessions. Using a combination of computer vision techniques and machine learning models, such as facial recognition, clustering, image processing, eye position estimation, and head pose estimation, our system is able to recognize and track individuals, enabling a better understanding of attentiveness during class. The use of advanced techniques like deep learning in the form of the '6DRepNet' model allows us to perform correct head pose estimation of a face, which helps us make the system more efficient in measuring attentiveness of students. The use of libraries like 'face recognition', 'scikit-learn', 'NumPy', 'OpenCV'[7] and 'MediaPipe' allowed us to implement different modules of the system in an efficient way. We developed an algorithm and named it **AttentionEstimator**, which made our system more efficient in measuring attentiveness. Despite the efficient measurement of individual attentiveness provided by the system, there are limitations that we aim to address in future iterations to further improve its usability. But overall, our system represents a valuable contribution to the field of online learning and engagement monitoring. The system's ability to track and monitor engagement levels can provide valuable insights to online tutors and educators, allowing them to adapt and adjust their teaching methods to better suit the needs of their students. Furthermore, the system can be used to monitor engagement levels in other video-based scenarios, making it a useful tool for a wide range of applications. The system is adaptable and can be integrated into various online tutoring platforms and applications. It can be extended to other video-based scenarios where engagement monitoring is important, such as in e-learning, teleconferencing, and remote collaboration.

Therefore, the system we designed can ultimately result in enhanced student learning experiences and higher tutoring session productivity. It helps in detection of the effectiveness of the teaching standards of the tutor and students during the tutoring session. High engagement levels can indicate a high quality of teaching. It can help to determine the appropriate compensation for the tutor. Additionally, our system enables the detection of the effectiveness of the learning standard of the pupil as well. It can help in providing valuable insights into the educational performance of the students, which can be shared with guardians to assist in identifying areas for improvement and providing support for the student's academic success. This can be used to identify any learning deficiencies and take appropriate measures to improve the student's performance. In general, this system offers a viable means of enhancing the online tutoring experience.

Bibliography

[1] M.-C. Su and C.-H. Chou, “A modified version of the k-means algorithm with a distance based on cluster symmetry,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 23, no. 6, pp. 674–680, 2001.

[2] A. Likas, N. Vlassis, and J. J. Verbeek, “The global k-means clustering algorithm,” *Pattern recognition*, vol. 36, no. 2, pp. 451–461, 2003.

[3] N. Gourier, D. Hall, and J. L. Crowley, “Estimating face orientation from robust detection of salient facial structures,” in *FG Net workshop on visual observation of deictic gestures*, Citeseer, vol. 6, 2004, p. 7.

[4] J. Shotton, A. Blake, and R. Cipolla, “Contour-based learning for object detection,” in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, IEEE, vol. 1, 2005, pp. 503–510.

[5] J. Smallwood, D. J. Fishman, and J. W. Schooler, “Counting the cost of an absent mind: Mind wandering as an underrecognized influence on educational performance,” *Psychonomic bulletin & review*, vol. 14, no. 2, pp. 230–236, 2007.

[6] E. Murphy-Chutorian and M. M. Trivedi, “Head pose estimation in computer vision: A survey,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 4, pp. 607–626, 2008.

[7] S. Brahmbhatt, *Practical OpenCV*. Apress, 2013.

[8] J. Cöster and M. Ohlsson, *Human attention: The possibility of measuring human attention using opencv and the viola-jones face detection algorithm*, 2015.

[9] J. Eriksson and L. Anna, *Measuring student attention with face detection:: Viola-jones versus multi-block local binary pattern using opencv*, 2015.

[10] O. M. Parkhi, A. Vedaldi, and A. Zisserman, “Deep face recognition,” 2015.

[11] R. Bixler and S. D'Mello, “Automatic gaze-based user-independent detection of mind wandering during computerized reading,” *User Modeling and User-Adapted Interaction*, vol. 26, no. 1, pp. 33–68, 2016.

[12] A. Gibaldi, M. Vanegas, P. J. Bex, and G. Maiello, “Evaluation of the tobii eyex eye tracking controller and matlab toolkit for research,” *Behavior research methods*, vol. 49, no. 3, pp. 923–946, 2017.

[13] N. Ruiz, E. Chong, and J. M. Rehg, “Fine-grained head pose estimation without keypoints,” in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 2074–2083.

[14] A. Ablavatski and I. Grishchenko, *Real-time ar self-expression with machine learning*, Mar. 2019. [Online]. Available: https://ai.googleblog.com/2019/03/real-time-ar-self-expression-with.html.

[15] M. Dewan, M. Murshed, and F. Lin, "Engagement detection in online learning: A review," *Smart Learning Environments*, vol. 6, no. 1, pp. 1–20, 2019.

[16] N. Veliyath, P. De, A. A. Allen, C. B. Hodges, and A. Mitra, "Modeling students' attention in the classroom using eyetrackers," in *Proceedings of the 2019 ACM Southeast Conference*, 2019, pp. 2–9.

[17] S. Peng, L. Chen, C. Gao, and R. J. Tong, "Predicting students' attention level with interpretable facial and head dynamic features in an online tutoring system (student abstract)," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 13 895–13 896.

[18] C. for Psychology in Schools and Education, *Managing attention and distractibility in online learning*, 2020. [Online]. Available: https://www.apa.org/topics/covid-%2019/managing-attention-distractibility-online-learning.

[19] A. Vakunov and D. Lagun, *Mediapipe iris: Real-time iris tracking depth estimation*, Aug. 2020. [Online]. Available: https://ai.googleblog.com/2020/08/mediapipe-iris-real-time-iris-tracking.html.

[20] M. Villa, M. Gofman, S. Mitra, A. Almadan, A. Krishnan, and A. Rattani, "A survey of biometric and machine learning methods for tracking students' attention and engagement," in *2020 19th IEEE international conference on machine learning and applications (ICMLA)*, IEEE, 2020, pp. 948–955.

[21] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun, "Repvgg: Making vgg-style convnets great again," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 13 733–13 742.

[22] H. Lee, M. Liu, H. Riaz, N. Rajasekaren, M. Scriney, and A. F. Smeaton, "Attention based video summaries of live online zoom classes," *arXiv preprint arXiv:2101.06328*, 2021.

[23] T. Hempel, A. A. Abdelrahman, and A. Al-Hamadi, "6d rotation representation for unconstrained head pose estimation," *arXiv preprint arXiv:2202.12555*, 2022.

[24] K. Roy and D. Chanda, "A robust webcam-based eye gaze estimation system for human-computer interaction," in *2022 International Conference on Innovations in Science, Engineering and Technology (ICISET)*, IEEE, 2022, pp. 146–151.