

POLITECNICO DI TORINO

Master's Degree in Computer Engineering



Master's Degree Thesis

A Feasibility Study for Enhancing Student's Engagement in Remote Lectures

Supervisors

Prof. Bartolomeo MONTRUCCHIO
Dr. Antonio Costantino MARCEDDU

Dr. Jacopo SINI

Dr. Luigi PUGLIESE

Candidate

Fereshteh FEIZABADI FARAHANI

December 2022

Summary

During the COVID-19 pandemic, almost all learning activities, as well as other daily face-to-face activities, were moved to online environments. This form allowed the students to continue almost all the usual educational activities that were normally carried out in school buildings, such as reading, writing, watching video tutorials, taking exams, and attending meetings.

Although e-learning has many advantages, it also has some disadvantages when compared to traditional face-to-face education. One of the most significant features is the lack of direct contact between teachers and students and less active participation with respect to face-to-face lectures. This led to the first new difficulties in deciphering common student states, such as boredom, confusion, and frustration. In this regard, technological support that can accurately and efficiently detect the level of attention of their students online can therefore be of great help to remote teachers. In this way, they could change their teaching approach dynamically and adopt techniques of capturing the attention of the students when necessary, increasing the effectiveness of the lessons themselves. So, e-learning platforms could enable the incorporation of novel technologies to estimate various factors such as eye blinking, gaze tracking, and facial expressions, which are important indicators of student engagement.

The purpose of this thesis is therefore to carry out a study on the possibility of creating an integrated system to detect the level of involvement and attention of students in real-time during remote lessons. The idea is that starting from the individual student video it is possible to detect input parameters such as gaze detection, blink detection, and facial expression detection.

There are several algorithms for detecting eye blinking and gazing. Three of these algorithms, called OpenFace, GazeTracking, and Gaze Controlled Keyboard were used to test videos shot in different conditions, including good and bad lighting and the presence or absence of glasses. These videos are representative of a normal lesson activity and involve viewing the webcam with simple eye movements to the right, left, and center and normal blinking. The accuracy of the blink and gaze tracking models was determined by comparing the output of the models to the ground truth. This was defined by manually labeling the individual video frames.

The comparison between the outputs of the models and the ground truth allowed us to select two of the models, which are OpenFace and GazeTracking, as the preferred ones for the realization of the system.

The OpenFace library is a comprehensive tool built with C++ for facial landmark detection, head pose estimation, facial action unit recognition, and eye-gaze estimation. It can do this by getting a video feed from a webcam or by using a prerecorded video. So, starting with the OpenFace GUI example provided with the library, facial expression detection was added to the software. The new C++ code is capable of detecting facial expressions using a trained Tensorflow model; it has been added as a static library to the OpenFace GUI, which was modified to show the expected emotion for each frame among the following: Angry, Contempt, Disgust, Fear, Happiness, Neutrality, Sadness, and Surprise.

GazeTracking is written in Python and requires the use of libraries such as NumPy, OpenCV, and Dlib. In this system, the Haar Cascade Classifier is used to detect facial landmarks on either a live webcam or recorded video. Thereafter, the library can detect the user's iris and provide labels such as blink, look right, left, and center. As for OpenFace, also for this library has been added the possibility of detecting facial expressions. The outputs of both programs are provided in real-time but are also saved in a CSV file to make a deferred analysis of the results.

We discovered that, in addition to facial expressions, the frequency of blinking and gaze tracking might be important factors in determining whether a student is engaged or distracted during remote lectures.

Table of Contents

List of Tables	VI
List of Figures	VII
1 Introduction	1
1.1 Background	1
1.2 What is Engagement in E-Learning Environments?	2
1.3 Our Idea	3
1.4 Thesis Structure	3
2 Extracting Features	5
2.1 Introduction	5
2.2 Face Detection	5
2.2.1 Challenges for Face Detection	6
2.3 Classical Algorithms of Face Detection	6
2.3.1 Haar Cascade	6
2.3.2 Dlib-HOG	7
2.4 Face Detectors Based on Deep Learning	7
2.4.1 SSD	7
2.4.2 MTCNN	8
2.4.3 Dual Shot Face Detector	8
2.4.4 RetinaFace	9
2.4.5 MediaPipe	10
2.4.6 YuNet	10
2.5 Performance Comparison of Face Detectors	11
2.5.1 Average Precision (AP)	11
2.6 Facial Landmarks	11
2.6.1 Dlib's Facial Landmark Detector	13
2.6.2 Understanding Dlib's Facial Landmark Detector	13
2.7 Blink Detection	14
2.7.1 Understanding the Eye Aspect Ratio	14

2.7.2	Real-Time Eye Blink Detection Using Facial Landmarks . . .	15
2.8	Studied Frameworks	15
2.8.1	Model 1: OpenFace	15
2.8.2	OpenFace Pipeline	16
2.8.3	Eye Gaze Estimation in OpenFace	17
2.8.4	Facial Expression Recognition in OpenFace	18
2.8.5	OpenFace Interface	18
2.8.6	OpenFace Code Layout	18
2.8.7	Model 2: Gaze Controlled Keyboard	22
2.8.8	Model 3: Eyes Position Estimator with MediaPipe	23
2.8.9	Model 4: Gaze Tracking	24
3	Test and Result of Models	27
3.1	Recorded Sample Videos	27
3.1.1	Data Logging	27
3.1.2	Ground Truth Annotation	28
3.1.3	Processing Data in Excel	29
3.2	Frameworks Outputs	30
3.2.1	Classification Accuracy	30
3.2.2	Frameworks Accuracy	30
3.2.3	OpenFace Output Format	30
3.2.4	Basic	32
3.2.5	Gaze Related	32
3.2.6	Action Units	32
3.3	Selected Frameworks	33
4	Emotion Detector Model	35
4.1	Facial Emotion Recognition	35
4.1.1	State of the Art	36
4.2	Choice of the Neural Network for Facial Expressions	36
4.2.1	Neural Networks Training	38
4.3	Facial Action Coding System	38
4.3.1	Intensity and Presence of AUs in OpenFace	39
4.3.2	Can AUs be used to detect facial emotions?	39
4.4	Integrated System	40
4.4.1	Result of Emotion Detector	42
4.4.2	Integrating Emotion Detector model with OpenFace	42
5	Detecting Engagement	45
5.1	Engagement in Remote Lectures	45
5.2	Improvements towards an Attention Detection Algorithm	46

5.3	Experiment	47
5.3.1	Application to Measure the Response Time	47
5.3.2	Our choice for Software	47
5.3.3	Standard Deviation	48
5.4	Experimental Results	48
5.4.1	Gaze Tracking	48
5.4.2	Facial Emotion	50
5.4.3	Data Processing Tools	50
6	Conclusions	55
6.1	Conclusions	55
6.2	Future Works	55
	Bibliography	57

List of Tables

2.1	Comparison of the models, based on FPS. Data from [25]	12
2.2	Comparison of the models, based on average precision at IoU threshold of 0.5. Data from [25]	12
3.1	Video frames annotations of the CSV files	28
4.1	Available pictures for each emotion in the chosen databases [52] ^a <i>FER+</i> annotations	38

List of Figures

2.1	Timeline of the evolution of face detection algorithms. Figure from [25]	6
2.2	Haar Cascade Classifier. Figure from [27]	7
2.3	Dlib-HOG. Figure from [25]	7
2.4	Structure of SSD. Figure from [25]	8
2.5	MTCNN. Figure from [25]	9
2.6	Dual Shot Face Detector. Figure from [30]	9
2.7	Machine learning solutions by MediaPipe. Figure from [33]	10
2.8	The area under the Precision-Recall curve is defined as the average precision.	11
2.9	Output of Dlib’s face landmarks on the images of HELEN dataset. Figure from [37].	13
2.10	The 68-face landmark coordinates from the iBUG 300-W dataset are graphically represented.	14
2.11	For multiple video sequence frames, the eye aspect ratio EAR is plotted. There is only one blink. Figure from [38]	15
2.12	OpenFace 2.0 implements modern facial behavior analysis algorithms, including facial landmark detection, head pose tracking, eye gaze, and facial action unit recognition. Figure from [39]	16
2.13	The OpenFace 2.0 facial behavior analysis pipeline includes landmark detection, head posture estimation, eye gaze estimation, and recognition of facial action units. All these systems’ outputs (shown in green) can be saved to disk or transmitted in real-time via the network. Figure from [39]	17
2.14	Example landmark detection from OpenFace 2.0. It is possible to notice its capability to deal with profile faces and occlusions. Figure from [39]	17
2.15	List of AUs in OpenFace 2.0. It predicts the intensity and presence of all AUs, except for AU28, for which only presence predictions are made. Figure from [39]	19

2.16	OpenFaceOffline running an analysis	20
2.17	OpenFace in Visual studio 2017	21
2.18	Head pose live	21
2.19	OpenFace demo	22
2.20	The facial landmarks for detecting the eyes through Dlib. Figure from [45]	23
2.21	Lines for detecting open and closed eyes. Figure from [45]	23
2.22	Screenshot of Eyes Position Estimator with MediaPipe	24
2.23	Screenshot of Gaze Tracking	24
2.24	Steps of eye isolation in GazeTracking. Figure from [47]	26
2.25	Impact of thresholding for detecting eyes. Figure from [47]	26
3.1	The output of all models with ground truth for calculating accuracy	28
3.2	Daylight	29
3.3	With glasses	29
3.4	Low light	29
3.5	The recorded videos in three different conditions	29
3.6	Considered states for eyes and looking	29
3.7	Accuracy of frameworks for gaze tracking and blinking	30
3.8	Value of gaze-angle-x column(looking right, left, center) in OpenFace for frequent blinking video	31
3.9	OpenFace output for gaze-angle-x versus ground truth	31
3.10	GazeTracking output for looking versus ground truth	31
3.11	Eye landmarks. Figure from [44]	33
3.12	CSV output of OpenFace	33
4.1	AUs. Figure from[63]	39
4.2	Presence and intensity of AUs Detected by OpenFace	40
4.3	The emotion and gaze direction of the subject are displayed by the GazeTracking application.	41
4.4	Structure of the Emotion Detector code. Figure from [64]	41
4.5	Visualization of the Tensorflow model for detecting facial expression by the Netron app	43
4.6	Frequency of Emotions in four sample recorded videos	44
4.7	Live webcam facial emotion detector in OpenFace	44
5.1	The workflow of the studied approach. Figure from [24]	45
5.2	Running the OpenFace on the three recorded videos	49
5.3	From the charts displayed, it is possible to notice that (a) is aligned with the ground truth of engagement (b)	50
5.4	Python code for computing the standard deviation on an input column	51
5.5	standard deviation for gaze value in sample video 1	52

5.6	standard deviation for gaze value in sample video 2	52
5.7	standard deviation for gaze value in sample video 3	53
5.8	Frequency of facial emotions in three samples; The dominant emotion is Neutral.	53
5.9	Standard deviation of Blinking for three samples	54
5.10	The sum of standard deviations for blinking in comparison to the ground truth	54

Chapter 1

Introduction

1.1 Background

In recent decades, E-learning and virtual education platforms have grown rapidly [1], becoming an essential component of the academic strategy of the most important higher education institutions in the world. During the COVID-19 outbreak, practically all learning and meeting activities were moved to online environments [2]. Online students participate in a variety of educational activities, such as reading, writing, watching video lectures, taking online exams, and attending online meetings. Participants in these educational activities indicate various levels of engagement, such as boredom, confusion, and frustration [3]. To give feedback to both instructors and students, online educators must precisely and efficiently detect their online learners' involvement status. For example, the teacher can adapt and make lectures more interesting by asking questions to non-interacting students to enhance their engagement. Since students in e-learning environments do not speak most of the time, engagement detection algorithms could be used to extract valuable information from solely visual input [4]. Such an operation is non-trivial and subjective because annotators can perceive varying levels of engagement from the same input video. Although e-learning has numerous advantages [5] it also has several disadvantages when compared to traditional face-to-face education. An example is an often reduced interaction between professors and students.

As a result, new e-learning platforms [6] enable the integration of novel technologies to determine various features such as attention level [7], heart rate [8], emotional state [9], gaze tracking, and head pose [10]. Among these behavioral modalities, estimating the level of attention of the students is particularly interesting for e-learning platforms [11].

This information could be used, for example, to: i) adapt dynamically to the environment and content [12] based on the attention level of the user, and ii)

improve the educational materials and resources with a posterior analysis of the e-learning sessions, e.g. detecting the most appropriate types of contents for a specific student and adapting the general information to them [13].

On the other hand, since the '70s, several studies have analyzed the relationship between the eye blink rate and the cognitive activity of the person such as the attention level [14] [15]. These studies suggest that lower eye blink rates can be associated with high attention periods while higher eye blink rates are related to lower attention levels. Consequently, eye blink detection can be a very useful tool to estimate the students' attention level and for improving e-learning platforms. Furthermore, eye blink detection can be used for fraud/cheating/lies detection as the eye blink rate decreases when cognitive demand increases [16].

There are two types of eye blink detection methods:

- Image-based approaches: in this case, the methods classify each individual frame as open, closed, or the degree of eye closure [17] [18].
- Video-based: these methods take into account the temporal information collected from the full sequence of frames [19] [20].

1.2 What is Engagement in E-Learning Environments?

Engagement is one of the qualitative measures in the learning process [21]. During learning it assumes a three-dimensional structure [22] including:

1. behavioral engagement such as staying on task.
2. emotional engagement such as boredom.
3. cognitive engagement such as focused attention [21].

All of these variables are crucial in determining levels of engagement. Three prominent approaches for measuring student engagement are self-reports, observational checklists, and automated assessments [21]. Both self-reports and observational checklists are still quite basic and unsuitable for real-time learning systems. Engagement may now be measured automatically through affective computing by detecting students' emotions. Smart systems for detecting students' emotions have been developed in the topic of affective computing. Some academics emphasize the importance of monitoring students' emotions and how these might be identified as a reflection of student participation [21]. Automated measurements are based on physiological and neurological sensor readings such as electroencephalography (EEG), heart rate, and so on [23]. They can also rely on computer vision techniques

such as facial expressions, gaze tracking, blinking, and so on. Based on our findings, there will be chances to improve the accuracy of engagement measurements when we employ more than one feature. Due to this reason, we suggest a new approach that requires inexpensive equipment, works in real-time, and is based on established technologies. It is based on the detection of the following features:

1. **Gaze tracking.**
2. **Blinking frequency.**
3. **Facial expressions.**

1.3 Our Idea

As mentioned earlier, facial expressions, blinking frequency, and gaze tracking are key elements in determining whether or not a student is interested in the remote learning topic. Therefore, in this thesis, we mainly focused on these features. In terms of facial expression detection, a Convolutional Neural Network (CNN) is used to recognize eight emotions that are now regarded as uniform across cultures: *anger, contempt, disgust, fear, happiness, neutrality, sadness, surprise*. The proposal consists of a system capable of accepting students' videos as input or a real-time stream via webcam and detecting their facial expressions, measuring their blinking frequency, and tracking their gaze. Based on these analyses, it can assess whether the student is engaged or distracted.

This study aims to improve the work made in [24] by including a gaze tracking and eye blink detection to evaluate if they are essential factors for detecting whether the student attending the remote lecture is focused or distracted.

1.4 Thesis Structure

The thesis is organized as follows:

- In **Chapter 2** we will review the various techniques to face detection, landmark detection, and blink detection, and then we introduce the frameworks we discovered for gaze tracking and blink detection.
- In **Chapter 3** we will talk about the recorded sample videos for testing and the output of the frameworks.
- In **Chapter 4** we will talk about the Facial Emotion Detector and Action Units (AU).

- In **Chapter 5** we will briefly review Engagement and our system outcomes on the long recorded videos.
- In **Chapter 6** we will wrap up the work and discuss possible future developments.

Chapter 2

Extracting Features

2.1 Introduction

In this chapter, we will first discuss various approaches to Face Detection in Section 2.2, followed by Facial Landmarks Detectors in Section 2.6. Then we'll look into Blink Detection in Section 2.7, and ultimately at Studied Frameworks in Section 2.8 that handle both gaze tracking and blinking.

In particular, during this thesis, 4 models have been implemented from the software available in the literature:

- OpenFace in Section 2.8.1.
- Gaze Controller Keyboard in Section 2.8.7.
- Eye Position Estimator with MediaPipe in Section 2.8.8.
- Gaze Tracking in Section 2.8.9.

2.2 Face Detection

Face detection is a subset of computer vision technologies that enables computers to recognize and locate human faces in images or video streams. This is the first and most essential stage in most computer vision applications that involve a face. An excellent facial detector is a necessary starting point for numerous activities involving the face, such as facial landmark detection, gender classification, attendance, crowd analysis, face tracking, and face recognition. In our study, we need to focus on landmark detection.

2.2.1 Challenges for Face Detection

Occlusion, lighting, skin color, posture, facial expression, sunglasses, wearing masks, and other factors can hinder the performance of a Face Detector. Since the students in our study are in a constrained environment and in front of the laptop, we evaluated a few of these obstacles in our recorded videos for detecting the face, such as whether the student is wearing glasses and is in a low-light environment.

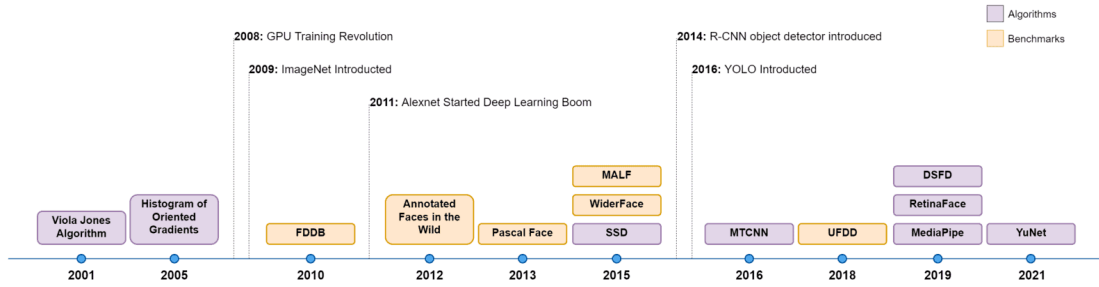


Figure 2.1: Timeline of the evolution of face detection algorithms. Figure from [25]

2.3 Classical Algorithms of Face Detection

2.3.1 Haar Cascade

In 2001, researchers Paul Viola and Michael Jones introduced the ViolaJones face detector [26], one of the first significant breakthroughs in this field. By utilizing the line or edge-detection features proposed in the ViolaJones detector, Haar Cascades provided a breakthrough in facial detection. Though it considerably improved detection speed and accuracy, it had limitations and failed when required to detect faces in noisy images.

There have been numerous advancements over the years. The Haar Cascade method uses for both face detection and eye detection. The classifier analyzes the pixel intensities and attempts to detect numerous specified features in the image. It can be certain that an object exists if it finds a sufficient number of matches for a certain region. Using OpenCV's Cascade Classifier function, we can easily import the available Haar Cascade Classifier XML files.

For more information, check the Haar-cascade Detection in OpenCV documentation. [27].

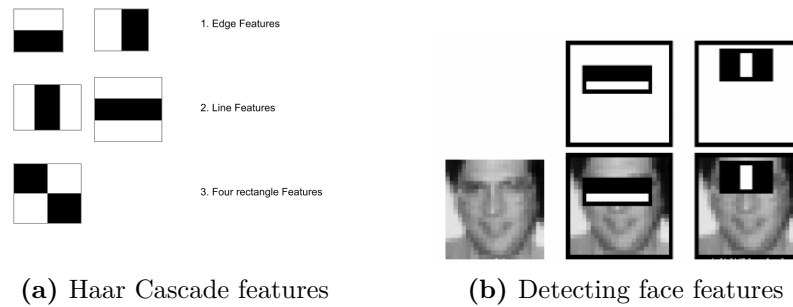


Figure 2.2: Haar Cascade Classifier. Figure from [27]

2.3.2 Dlib-HOG

Dlib is a popular Face Detector that combines the standard Histogram of Gradients (HoG) feature with a linear classifier model, an image pyramid, and a sliding window detection approach. Dlib employs five HOG filters: front-facing, left-facing, right-facing, front-facing but rotated left, and front-facing but rotated right.

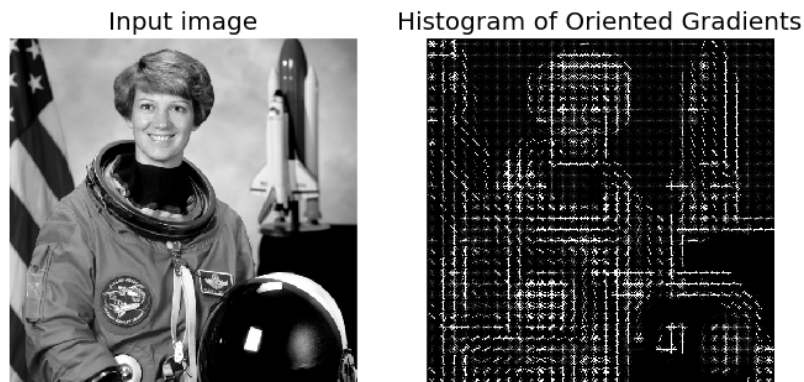


Figure 2.3: Dlib-HOG. Figure from [25]

2.4 Face Detectors Based on Deep Learning

2.4.1 SSD

Single Shot MultiBox Detector [28] name reveals most of the details about the model. It finds the object in a single pass through the input image, as opposed to other models that traverse the image numerous times to produce an output

detection.

The Backbone model is a feature map extractor that is pre-trained image classification. The SSD head is made up of stacked convolutional layers and is inserted to the top of the backbone model. This produces the result in the form of bounding boxes over the objects. The numerous items in the image are detected by these convolutional layers.

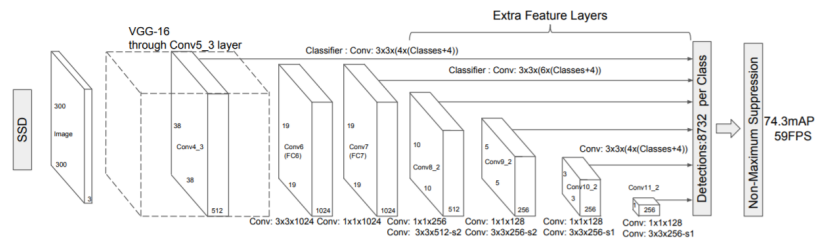


Figure 2.4: Structure of SSD. Figure from [25]

2.4.2 MTCNN

MTCNN, or Multi-Task Cascaded Convolutional Neural Network. This widely used model, published in 2016, is made up of neural networks linked in a cascade pattern.[29]

The proposed MTCNN architecture is made up of three CNN phases. The first step, P-Net (Proposal Network), uses a shallow CNN to generate candidate windows quickly. The windows are then refined in the R-Net (Refine Network) stage by rejecting several non-face bounding boxes using a more complicated CNN. Finally, the O-Net (Output Network) stage refines the result with a more robust CNN and outputs five facial landmark positions.[29]

2.4.3 Dual Shot Face Detector

Dual Shot Face Detector is an innovative Face Detection technique that addresses the three primary areas of Facial Detection: Improved feature learning, progressive loss design, and anchor assign-based data augmentation are all examples of improvements.[30]

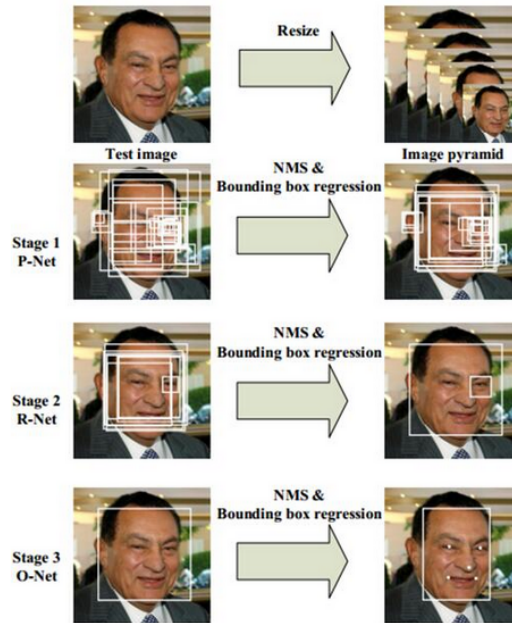


Figure 2.5: MTCNN. Figure from [25]

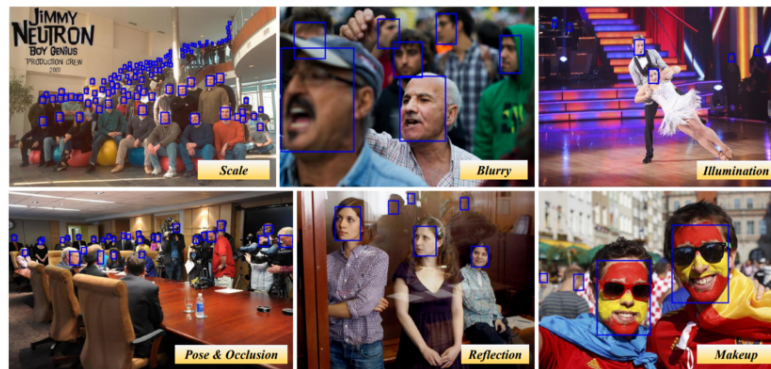


Figure 2.6: Dual Shot Face Detector. Figure from [30]

2.4.4 RetinaFace

RetinaFace is a practical single-stage state-of-the-art face detector that is part of the InsightFace project from DeepnInsight, which is also credited with other top Face-Recognition techniques such as ArcFace, SubCenter ArcFace, PartialFC, and other facial applications.[31]

2.4.5 MediaPipe

Face Detection by MediaPipe is an ultrafast face detection system with 6 landmarks and multi-face support. It is built on BlazeFace [32], a lightweight and high-performance face detector designed specifically for mobile GPU inference.

Because of super-realtime performance of the detector, it can be used in any live viewfinder experience that requires an accurate facial region of interest as an input for other task-specific models like 3D facial keypoint estimation (e.g., MediaPipe Face Mesh), facial features or expression classification, and face region segmentation.[33]

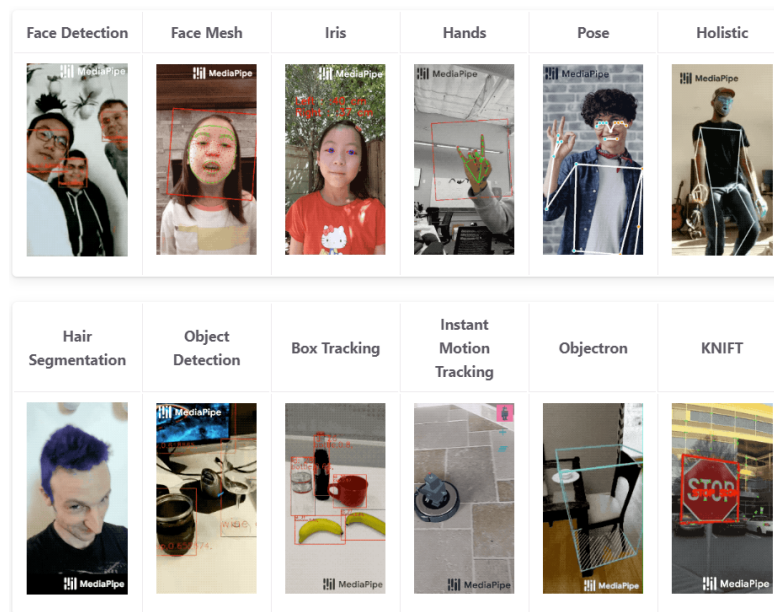


Figure 2.7: Machine learning solutions by MediaPipe. Figure from [33]

2.4.6 YuNet

Traditionally, OpenCV was equipped with face detectors such as Haar cascades and HOG detectors, which performed well for frontal faces but failed in other situations. This difficulty was solved in the most current release of OpenCV (4.5.4 Oct 2021), which included a face detection model called YuNet.

It is a face detector based on CNN. It is a lightweight and quick model. It may be installed on nearly any device with a model size of less than one megabyte. It uses MobileNet as its backbone and has a total of 85000 parameters. [34]

2.5 Performance Comparison of Face Detectors

2.5.1 Average Precision (AP)

A measure called Average Precision is used to assess the efficacy of object detection and localization algorithms (AP). Choosing a confidence value for your application can be hard and subjective. The area under the PR curve defines *Average precision*, an important performance metric that tries to reduce the dependency on selecting a single confidence threshold value. AP transforms the PR Curve to a single scalar value. When both precision and recall are high, the average precision is high; when one is poor, the average precision is low across a variety of confidence threshold values. AP has a value between 0 and 1 [35].

$$AveragePrecision(AP) = \int_{r=0}^1 p(r)dr$$

Figure 2.8: The area under the Precision-Recall curve is defined as the average precision.

The tables below 2.1 2.2 compare all of the previous Face-Detection models in terms of Average Precision (AP) and inference speed in Frames Per Second (FPS).

System Configuration:

- Processor: Intel(R) Xeon(R) CPU.
- Operating speed: 2.20 GHz.
- Total RAM: 12.69 GB.

The authors used the Tesla P100-16GB GPU on the Google Colab.[25]

2.6 Facial Landmarks

Facial landmarks are used to identify and portray prominent facial features such as the eyes, brows, nose, mouth, and jawline. The challenge of detecting facial landmarks is a sub task of the shape prediction. A shape predictor attempts to localize important areas of interest across an input image and normally a ROI (Region Of Interest) that specifies the object of interest.

Model	FPS(Colab GPU)	FPS(Colab CPU)
Haar cascade	–	19.95
Dlib	–	33.92
SSD	19.90	15.58
MTCNN	2.11	1.81
MediaPipe	323.63	225.34
RetinaFace Resnet50	72.24	1.43
RetinaFace MobilenetV1	69.50	28.89
Dual Shot Face Detector	18.89	0.22
YuNet	–	49.43

Table 2.1: Comparison of the models, based on FPS. Data from [25]

Model	AP@0.5
SSD	0.931
MTCNN	0.915
MediaPipe	0.743
RetinaFace Resnet50	0.994
RetinaFace MobilenetV1	0.994
Dual Shot Face Detector	0.989
YuNet	0.994

Table 2.2: Comparison of the models, based on average precision at IoU threshold of 0.5. Data from [25]

Our goal is to detect important facial features on the face using shape prediction methods. Detecting facial landmarks is a two-step process:

- Step 1: Localize the face in the image.
- Step 2: Detect the key facial structures on the face ROI.

Face detection **Step1** : can be achieved through one of the methods explained before. We could use OpenCV’s built-in Haar cascades. We might apply a pre-trained HOG + Linear SVM object detector specifically for face detection. Alternatively, we might even use deep learning-based algorithms for face localization. In any case, what matters is that we extract the face bounding box (i.e., the (x, y)-coordinates of the face in the image). Given the face region, we can then apply **Step 2**: detecting key facial structures in the face region, that in our case is the eyes.

2.6.1 Dlib’s Facial Landmark Detector

Dlib is a modern C++ toolkit that contains machine learning techniques and tools for developing complex C++ software to deal with real-world challenges [36]. The facial landmark detector in the Dlib package is an implementation of this work: one-millisecond face alignment with an ensemble of regression trees [37].

This method starts by using:

1. A training set of labeled face landmarks on an image. These images are manually labeled, with particular (x, y)-coordinates of regions surrounding each face component specified.
2. Priors, or more specifically, the probability of the distance between pairs of input pixels.

Given this training data, an ensemble of regression trees is trained to estimate the facial landmark positions directly from the pixel intensities (i.e., no “feature extraction” is taking place). As a result, a facial landmark detector with high-quality predictions can be utilized to detect facial landmarks in real-time; see this as an example 2.9.



Figure 2.9: Output of Dlib’s face landmarks on the images of HELEN dataset. Figure from [37].

2.6.2 Understanding Dlib’s Facial Landmark Detector

The pre-trained facial landmark detector in the Dlib package estimates the position of 68 coordinates that match to facial structures on the face. These indexes of the 68 coordinates can be visualized in Figure 2.10.

These annotations are part of the 68-point iBUG 300-W dataset, which was used to train the Dlib facial landmark predictor. Other face landmark detectors exist, such as the 194-point model that can be trained on the HELEN dataset. The same Dlib framework may be used to train the shape predictor on the input training

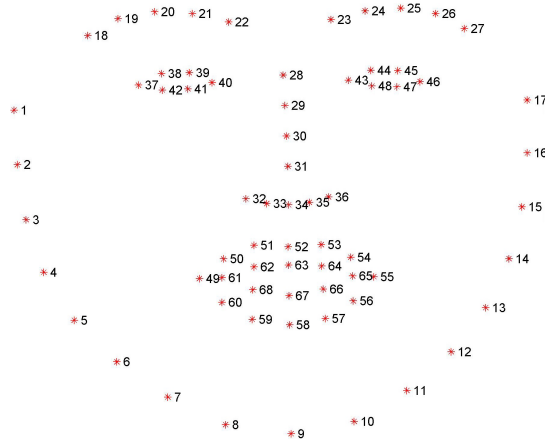


Figure 2.10: The 68-face landmark coordinates from the iBUG 300-W dataset are graphically represented.

data regardless of the dataset - helpful if you need to train your face landmark detectors or your shape predictor. Dlib’s 68-point facial landmark detector tends to be the most popular in computer vision due to its speed and reliability.

2.7 Blink Detection

2.7.1 Understanding the Eye Aspect Ratio

As previously stated, facial landmark detection can be used to locate important facial regions such as the eyes, brows, nose, ears, and mouth. This also suggests that by knowing the indexes of specific facial regions, we can extract certain facial structures. We are only interested in two sets of face components for blink detection: the eyes. Each eye is represented by six (x, y)-coordinates, starting at the left corner of the eye (as if looking at the person) and moving clockwise around the rest of the region. These coordinates have a relation between width and height. Where p_1, \dots, p_6 are 2D facial landmark locations.[38]

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

This equation’s numerator computes the distance between vertical eye landmarks. In contrast, the denominator computes the distance between horizontal eye landmarks, with the denominator correctly weighted because there is only one set of horizontal points but two sets of vertical points.

The eye aspect ratio (EAR) is about constant while the eye is open but rapidly drops to zero when the eye blinks. We can skip image processing procedures by

employing this simple equation and only rely on the ratio of ocular landmark distances to determine if a person is blinking.

2.7.2 Real-Time Eye Blink Detection using Facial Landmarks

The eye blink is a fast closing and reopening of a human eye. Each individual has a little bit different pattern of blinks. The pattern differs in the speed of closing and opening, the degree of squeezing the eye, and the blink duration. The eye blink lasts approximately 100-400 ms.

We use state-of-the-art facial landmark detectors to localize the eyes and eyelid contours. We derive the eye aspect ratio (EAR) from the landmarks detected in the image to estimate the eye-opening state. Since the per-frame EAR may not necessarily recognize the eye blinks correctly, a classifier that considers a larger temporal window of a frame is trained.

Each video frame has its eye landmarks recognized. The eye aspect ratio (EAR) is computed as the eye's height-to-width ratio. where p_1, \dots, p_6 are the two-dimensional landmark locations indicated in the figure 2.11. The EAR is mostly constant when an eye is open and gets close to zero while closing an eye.

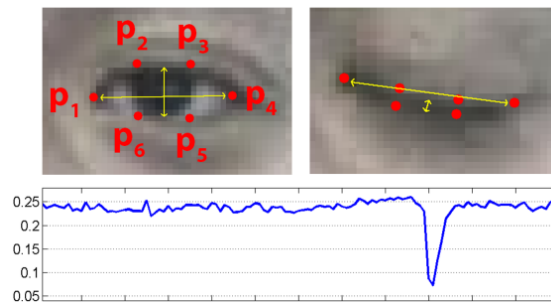


Figure 2.11: For multiple video sequence frames, the eye aspect ratio EAR is plotted. There is only one blink. Figure from [38]

2.8 Studied Frameworks

2.8.1 Model 1: OpenFace

OpenFace 2.0 is a tool for computer vision and machine learning researchers, the affective computing community, and anyone interested in developing interactive

apps based on facial behavior analysis. OpenFace 2.0 is an update of the OpenFace toolbox that can identify more accurate **facial landmarks**, **estimate head posture**, **recognize facial activity units**, and **estimate eye gaze** [39].

This tool is capable of real-time performance and can run from a simple webcam without any specialist hardware. Finally, the source code for training and running models is freely available for research purposes.

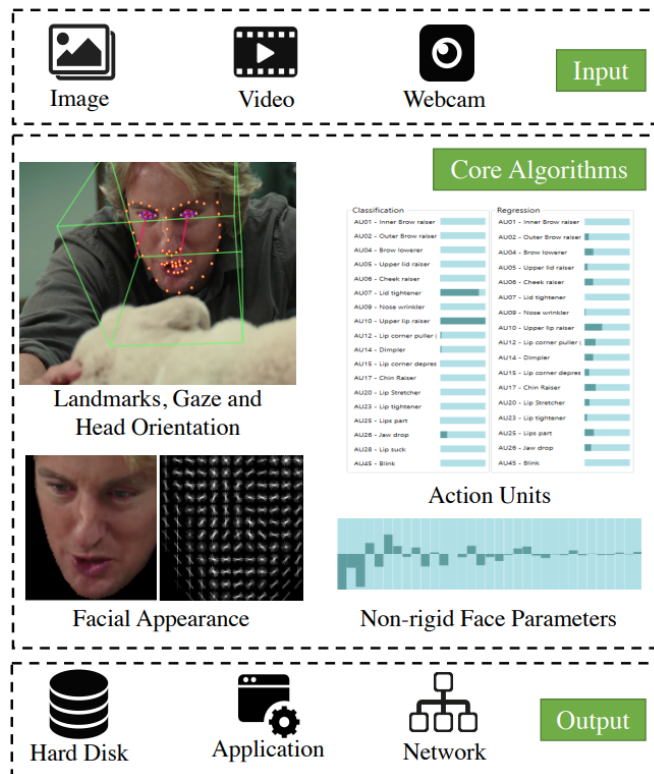


Figure 2.12: OpenFace 2.0 implements modern facial behavior analysis algorithms, including facial landmark detection, head pose tracking, eye gaze, and facial action unit recognition. Figure from [39]

2.8.2 OpenFace Pipeline

This section describes the fundamental technologies that OpenFace 2.0 use for facial behavior analysis. It uses Convolutional Experts Constrained Local Model (CE-CLM) [40] for facial landmark detection and tracking. Example landmark detections can be seen in Figure 2.14.

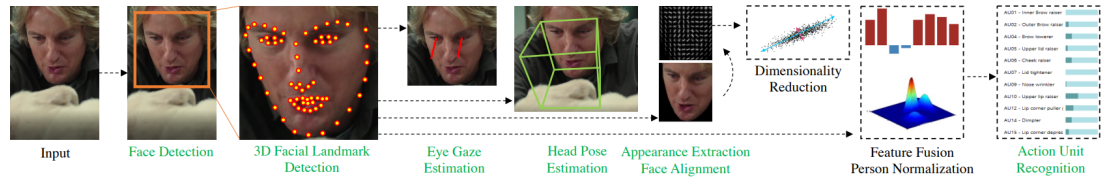


Figure 2.13: The OpenFace 2.0 facial behavior analysis pipeline includes landmark detection, head posture estimation, eye gaze estimation, and recognition of facial action units. All these systems’ outputs (shown in green) can be saved to disk or transmitted in real-time via the network. Figure from [39]

OpenFace 2.0 novelties: C++ implementation of CE-CLM in OpenFace 2.0 includes several speed optimizations that enable real-time performance. These include deep model simplification, multiple innovative hypotheses, and sparse response map computation.



Figure 2.14: Example landmark detection from OpenFace 2.0. It is possible to notice its capability to deal with profile faces and occlusions. Figure from [39]

2.8.3 Eye gaze Estimation in OpenFace

To estimate eye gaze, OpenFace2 uses a Constrained Local Neural Field (CLNF) landmark detector [41] to detect eyelids, iris, and the pupil. For training the landmark detector, it used the SynthesEyes training dataset [42]. It uses the detected pupil and eye location to compute each eye’s gaze vector individually.

2.8.4 Facial Expression Recognition in OpenFace

OpenFace 2.0 detects facial expressions by measuring the intensity and presence of facial action units (AU). It employs an approach based on a recent AU identification framework developed by Baltrusaitis et al. [43] which employs linear kernel Support Vector Machines. OpenFace 2.0 includes a direct implementation with a few changes that allow it to operate better on natural video sequences by employing person-specific normalization and prediction correction. While this may appear to be a simple and out-of-date model for AU recognition at first glance, the OpenFace experiment shows that it is competitive, even compared to newer deep learning algorithms, while maintaining a significant speed advantage. Facial expression recognition models are trained on various datasets where the AU labels overlap. As a result, OpenFace 2.0 recognizes the AUs listed 2.15. For our research, we need to read the AU45 value when the eyes blink.

2.8.5 OpenFace Interface

OpenFace 2.0 is an easy-to-use toolbox for the analysis of facial behavior. There are two main ways of using OpenFace 2.0: Graphical User Interface (for Microsoft Windows) and command line (for Microsoft Windows, Ubuntu, and Apple macOS). As the source code is available, it is also possible to integrate it into any C++, C, or Matlab-based project.

To make the system easier to use, they provide sample Matlab scripts demonstrating how to extract, save, read and visualize each behavior. It can operate on real-time data video feeds from a webcam, recorded video files, image sequences, and individual images. It is possible to save the outputs of the processed data as CSV files in case of facial landmarks, shape parameters, head pose, action units, and gaze vectors.[39] Check out the GitHub repository of the project for details on how to install, compile, and use it[44].

2.8.6 OpenFace Code Layout

The OpenFace software layout is described in this section.

- **Libraries:**
 - Local: The actual section of code that contains the relevant computer vision algorithms.
 - CppInterop: Wrappers around C++ code for C sharp bindings.
 - LandmarkDetector: The CE-CLM, CLNF, and CLM algorithms together with face tracking code.



















AU	Full name	Illustration
AU1	INNER BROW RAISER	
AU2	OUTER BROW RAISER	
AU4	BROW LOWERER	
AU5	UPPER LID RAISER	
AU6	CHEEK RAISER	
AU7	LID TIGHTENER	
AU9	NOSE WRINKLER	
AU10	UPPER LIP RAISER	
AU12	LIP CORNER PULLER	
AU14	DIMPLER	
AU15	LIP CORNER DEPRESSOR	
AU17	CHIN RAISER	
AU20	LIP STRETCHED	
AU23	LIP TIGHTENER	
AU25	LIPS PART	
AU26	JAW DROP	
AU28	LIP SUCK	
AU45	BLINK	

Figure 2.15: List of AUs in OpenFace 2.0. It predicts the intensity and presence of all AUs, except for AU28, for which only presence predictions are made. Figure from [39]

- FaceAnalyser: Facial Action Unit detection and some useful code for extracting features for facial analysis.
- GazeAnalyser: Code for extracting eye gaze.
- Utilities: Utility code for image and video reading, result recording, result visualization, and rotation and image manipulation.
- 3rdParty: A place for 3rd party libraries.
- CameraEnumerator: Useful utility for listing and naming connected webcams
- OpenCV3.4: Prepackaged OpenCV 3.4 library that is used extensively internally to provide support for basic computer vision functionality.

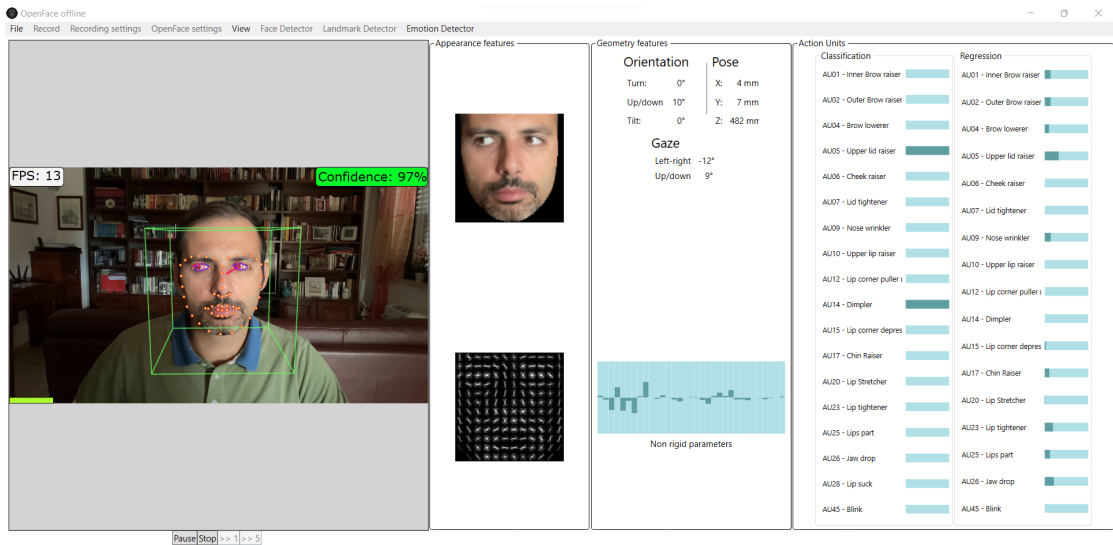


Figure 2.16: OpenFaceOffline running an analysis

- dlib: A header only dlib library (includes the face detector used for in-the-wild images).
- OpenBLAS: Prepackaged OpenBLAS code for fast matrix multiplication
- **Executable:** The runner and executables that show how to use the libraries for facial expression and head pose tracking.
 - FeatureExtraction: Main workhorse executable for sequences; extracting all supported features from faces: landmarks, AUs, head pose, gaze, similarity normalized faces and HOG features.
 - FaceLandmarkImg: Extraction of all supported features from faces: landmarks, AUs, head pose, gaze, similarity normalized faces, and HOG features.
 - FaceLandmarkVid: Running single-person landmark detection and gaze extraction on videos on disk or from a webcam.
 - FaceLandmarkVidMulti: Tracking multiple faces in sequences and extracting their features (from a webcam, video file, or image sequence).
 - releases: Scripts for packaging the Windows binaries into a release.
- **GUI:** the Windows GUI executables
 - HeadPose live: Tracking and recording head pose.
 - OpenFaceDemo: Nice visualization of OpenFace results from a webcam.

Extracting Features

- OpenFaceOffline: GUI for processing videos, image sequences, collections of images, and webcam feeds, same functionality as that of FeatureExtraction.

The screenshots of the GUI are shown below; for our experiment, we utilized the OpenFaceOffline executable, which has all of the functionality we require, such as preserving the analysis results and supporting the import of recorded videos.

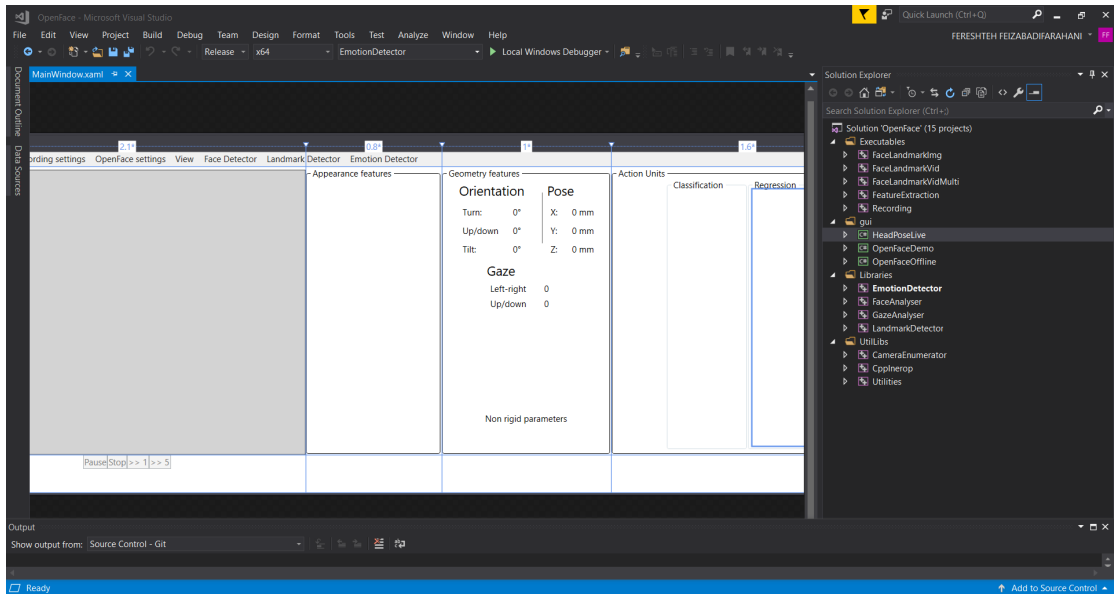


Figure 2.17: OpenFace in Visual studio 2017

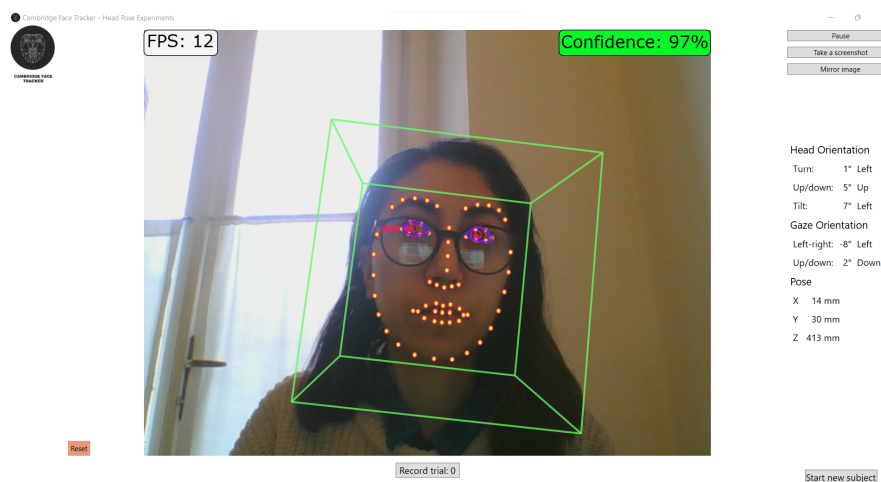


Figure 2.18: Head pose live

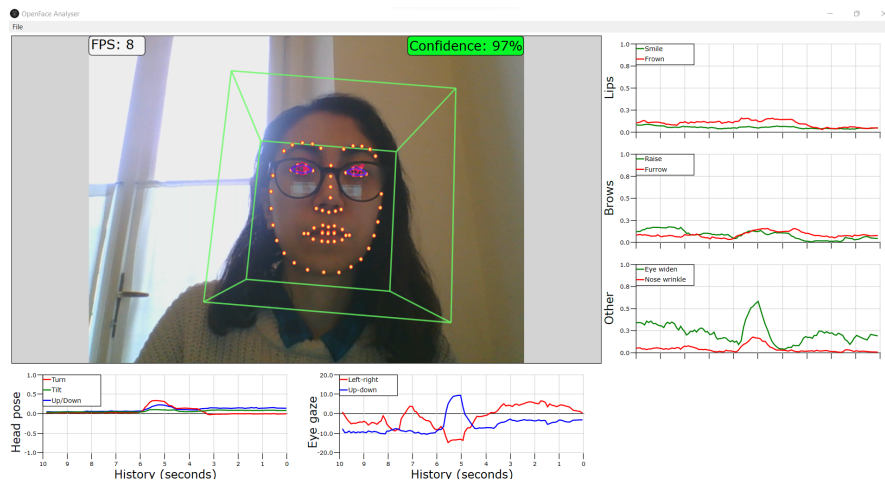


Figure 2.19: OpenFace demo

2.8.7 Model 2: Gaze Controlled Keyboard

This project **Gaze Controlled Keyboard** is a part of this project: [45] written in Python, which aims to create a virtual keyboard that can be controlled by eye tracking and blinking. The Opencv, Numpy, and Dlib libraries are used for face detection and eye localization in real-time camera frames.

We used part 4 to track the eye gazing to the right, left, center, and blinking. We stored the output of each frame in a CSV file for later study. Using the face landmarks detection approach, we can find 68 specific landmarks of the face. To each point, a specific index is assigned. We need two detect the two eyes separately:

- Left eye points: (36, 37, 38, 39, 40, 41)
- Right eye points: (42, 43, 44, 45, 46, 47)

An eye is blinking when:

- The eyelid is closed.
- We can not see the eyeball anymore.
- The bottom and upper eyelashes connect.

Also, remember that all these activities must occur in a short period (a blink of an eye takes around 0.3 to 0.4 seconds); otherwise, the eye is just closed.

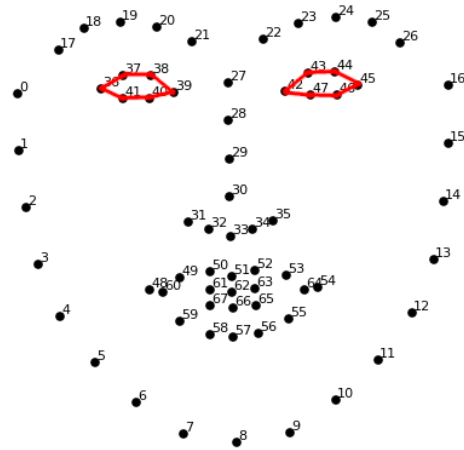


Figure 2.20: The facial landmarks for detecting the eyes through Dlib. Figure from [45]

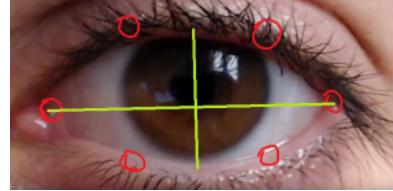


Figure 2.21: Lines for detecting open and closed eyes. Figure from [45]

2.8.8 Model 3: Eyes Position Estimator with MediaPipe

This project **Eyes Position Estimator Mediapipe** [46] used the MediaPipe python library for detecting facial landmarks. It works through below steps:

1. Detecting facial landmarks with MediaPipe.
2. Identifying eye Landmarks.
3. Draw eyes, eyebrows, lips, and a face oval.
4. Recognizing the blinks.
5. Counting blinks.
6. Extracting eyeballs from the frame using masking techniques.
7. Thresholding eyes to distinguish between black and white pixels.
8. Separate each eye into three pieces (right piece, centerpiece, and left piece).
9. Counting and determining the position of black pixels.



Figure 2.22: Screenshot of Eyes Position Estimator with MediaPipe

2.8.9 Model 4: Gaze Tracking

Gaze tracking [47] is a webcam-based eye-tracking system. It provides the precise position of the pupils as well as the gaze direction, in real-time. It used NumPy, OpenCV, and Dlib libraries. Except for OpenFace, we have added the option to save the output to a CSV file for all the models we have chosen.

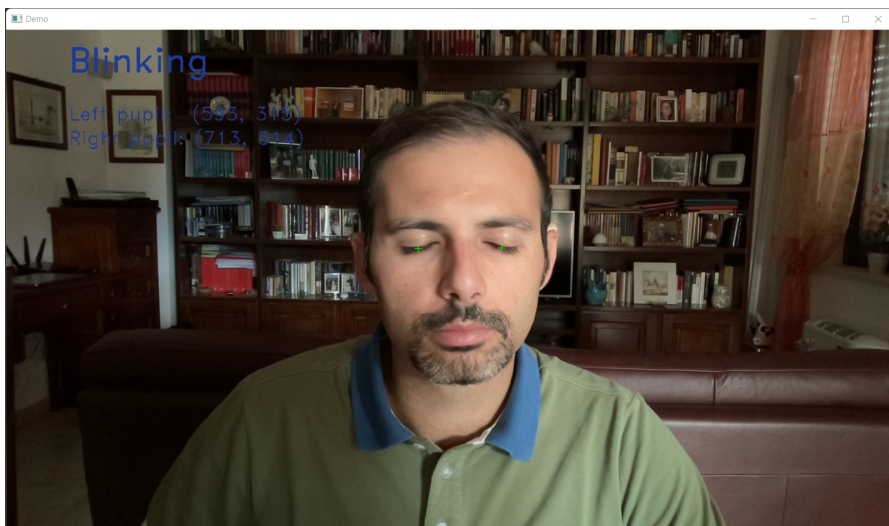


Figure 2.23: Screenshot of Gaze Tracking

The algorithm is simple and operates in the following steps:

- Separate the eyes from the rest of the face. This is accomplished by capturing

certain facial landmarks with `shape_predictor_68_face_landmarks.dat`; The process is performed using the `eye.py` file.

- Detect the iris using `pupil.py` contours.
- Use `gazetracking.py` to estimate the gaze position by measuring the iris coordinates.

These steps are used to isolate the eyeballs:

- Removing any noise by slightly blurring the image.
- Backlights are being eliminated from the image by degrading it.
- Binarize the image such that only black and white pixels remain (no grayscale).
- Determine the centroid by detecting the contour (the pupil position).

However, a threshold value should be applied when binarizing the image to distinguish between white and black pixels. This value varies greatly depending on the people and webcams (about 10 to 75). Pupil detection can be very accurate if the value is correct or wildly inaccurate if the value is incorrect. To avoid specifying a threshold, an automatic calibration method is implemented to determine the appropriate threshold value for the user/webcam.

The author of GazeTracking conducted some research and discovered that the iris is always around 48 percent of the eye's surface in all people (when they are looking at the center). Threshold levels for binarizing images might vary significantly from person to person, while iris diameters are generally stable.

The following is how automated calibration works:

- The frames are sent to the Calibration class during the first 20 frames.
- Determine iris sizes by binarizing the frame with different threshold values ranging from 5 to 100.
- The value that generates the closest iris size to 48 percent is stored for each frame.
- After evaluating the first 20 frames, the final threshold value is the average of the best 20 values [47].

In the next chapter, the outcome of the models for recorded videos will be discussed.

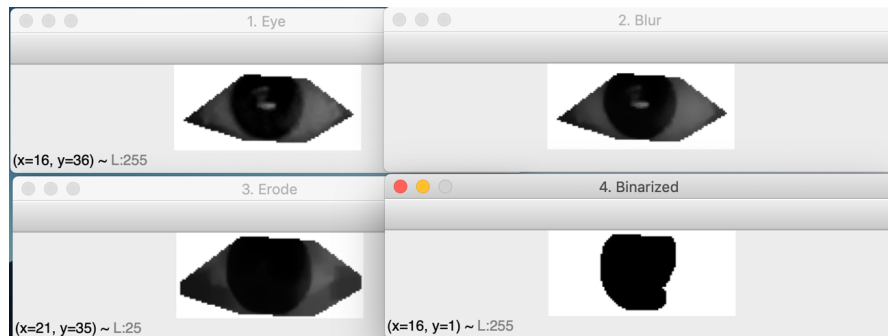


Figure 2.24: Steps of eye isolation in GazeTracking. Figure from [47]

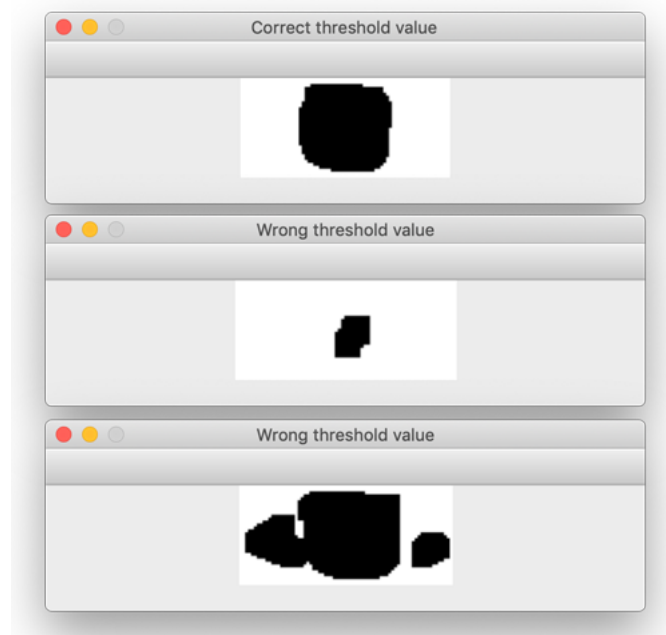


Figure 2.25: Impact of thresholding for detecting eyes. Figure from [47]

Chapter 3

Test and Result of Models

This chapter discusses the recorded sample videos for testing in Section 3.1 and annotation of the ground truth in Section 3.1.2, followed by the outputs of the frameworks in Section 3.2 and lastly, the selected frameworks in Section 3.3.

3.1 Recorded Sample Videos

The reliability of the selected frameworks was tested under four conditions in a constrained environment:

- Daylight environment.
- Low light environment.
- When the subject is wearing glasses.
- When the subject blinks more frequently.

Figure 3.5 represents these conditions. The ground truth was created by annotating manually, for each frame, the expected values in terms of eye-gazing and eye-blinking. Each value was written in a CSV file to calculate how accurate the selected frameworks are.

3.1.1 Data Logging

The log files generated by each one of the benchmarks contain information about the frame number within the analyzed video, timestamp, blinking state, and looking direction. Table 3.1 shows the corresponding state value for blinking and looking. These columns contain the output of the outputs of the framework, while others were added to allow us to compare the result of the frameworks to each other. Follows a numerical description of the content of the CSV file:

- Numerical value:
 - 0 if the person is looking to the center.
 - 1 if the person is looking to the right.
 - 2 if the person is looking to the left.
 - -1 if the person is blinking.
- Blinking accuracy: it compares if the value is equal to the ground truth (true) or not (false).
- Looking accuracy: it compares if the value is equal to the ground truth (true) or not (false).

Figure 3.1 shows the excel file with all these columns.

Looking state	Value
blink	-1
center	0
right	1
left	2

Table 3.1: Video frames annotations of the CSV files

Ground Truth				Model 1: OpenFace						Model 2: Gaze Tracking					
frame	timestamp	blink	looking	Numerical_Value	looking	gaze_angle_x	AU45_c	Accuracy for looking	Accuracy for Blink	blink	looking	Numerical_Value	Accuracy for Looking	Accuracy for Blinking	
1	0	0	center	0	0	center	-0.023	0	TRUE	TRUE	0	Looking center	0	TRUE	TRUE
2	0.033	0	center	0	0	center	-0.024	0	TRUE	TRUE	0	Looking center	0	TRUE	TRUE
3	0.067	0	center	0	0	center	-0.027	0	TRUE	TRUE	0	Looking center	0	TRUE	TRUE
4	0.1	0	center	0	0	center	-0.022	0	TRUE	TRUE	0	Looking center	0	TRUE	TRUE
5	0.133	0	center	0	0	center	-0.019	0	TRUE	TRUE	0	Looking center	0	TRUE	TRUE
6	0.167	0	center	0	0	center	-0.018	0	TRUE	TRUE	0	Looking center	0	TRUE	TRUE
7	0.2	0	center	0	0	center	-0.017	0	TRUE	TRUE	0	Looking center	0	TRUE	TRUE
8	0.233	0	center	0	0	center	-0.019	0	TRUE	TRUE	0	Looking center	0	TRUE	TRUE
9	0.267	0	center	0	0	center	-0.018	0	TRUE	TRUE	0	Looking center	0	TRUE	TRUE
10	0.3	0	center	0	0	center	-0.032	0	TRUE	TRUE	0	Looking center	0	TRUE	TRUE
11	0.333	0	center	0	0	center	-0.034	0	TRUE	TRUE	0	Looking center	0	TRUE	TRUE
12	0.367	0	center	0	0	center	-0.024	0	TRUE	TRUE	0	Looking center	0	TRUE	TRUE
13	0.4	0	center	0	0	center	-0.021	0	TRUE	TRUE	0	Looking center	0	TRUE	TRUE
14	0.433	0	center	0	0	center	-0.026	0	TRUE	TRUE	0	Looking center	0	TRUE	TRUE
15	0.467	0	center	0	0	center	-0.025	0	TRUE	TRUE	0	Looking center	0	TRUE	TRUE
16	0.5	0	center	0	0	center	-0.031	0	TRUE	TRUE	0	Looking center	0	TRUE	TRUE
17	0.533	0	center	0	0	center	-0.023	0	TRUE	TRUE	0	Looking center	0	TRUE	TRUE
18	0.567	0	center	0	0	center	-0.033	0	TRUE	TRUE	1	Blinking	-1	FALSE	FALSE
19	0.6	1	-	-1	-1	-	0.003	0	TRUE	FALSE	1	Blinking	-1	FALSE	TRUE
20	0.633	1	-	-1	-1	-	0.01	1	TRUE	TRUE	1	Blinking	-1	FALSE	TRUE
21	0.667	1	-	-1	-1	-	0.019	1	TRUE	TRUE	1	Blinking	-1	FALSE	TRUE
22	0.7	1	-	-1	0	center	-0.019	1	FALSE	TRUE	1	Blinking	-1	FALSE	TRUE
23	0.733	1	-	-1	0	center	-0.023	1	FALSE	TRUE	1	Blinking	-1	FALSE	TRUE
24	0.767	1	-	-1	0	center	-0.002	1	FALSE	TRUE	1	Blinking	-1	FALSE	TRUE
25	0.8	1	-	-1	-1	-	0.017	1	TRUE	TRUE	1	Blinking	-1	FALSE	TRUE
26	0.833	1	-	-1	-1	-	0.013	1	TRUE	TRUE	1	Blinking	-1	FALSE	TRUE
27	0.867	1	-	-1	-1	-	0.034	1	TRUE	TRUE	1	Blinking	-1	FALSE	TRUE
28	0.9	1	-	-1	-1	-	0.043	1	TRUE	TRUE	1	Blinking	-1	FALSE	TRUE

Figure 3.1: The output of all models with ground truth for calculating accuracy

3.1.2 Ground Truth Annotation

We extracted the frames of videos by using a Python script and we recorded the corresponding value for the gaze and blink state in the CSV by viewing each frame by hand. In our study we only considered looking left, center, and right as poses, Figure shows 3.6.



Figure 3.2: Daylight **Figure 3.3:** With glasses **Figure 3.4:** Low light

Figure 3.5: The recorded videos in three different conditions

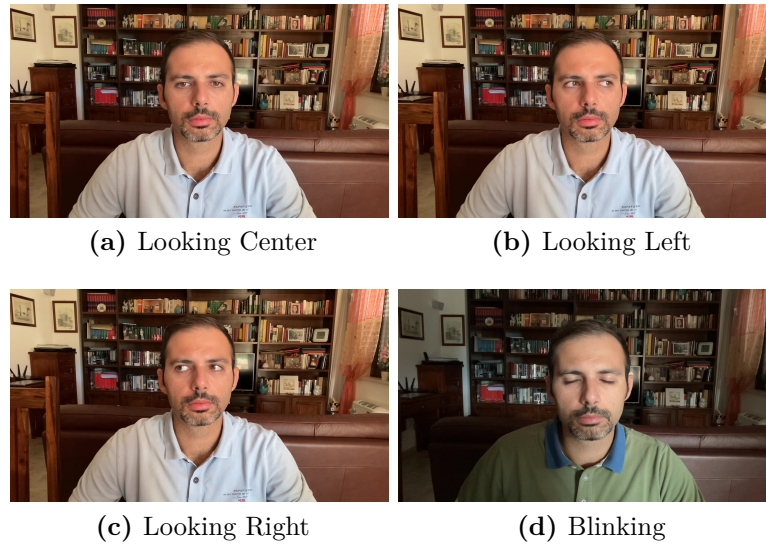


Figure 3.6: Considered states for eyes and looking

3.1.3 Processing Data in Excel

We used some formulas to unify the output of the models in Excel.

- **IF** is used for mapping the value of gaze-angle-x column to three labels: center, right, and left. The threshold values are the following: Center ≤ 0 and ≥ -0.05 , Right ≤ -0.2 , and Left ≥ 0.05 .
- **IFS** is used for mapping the categorical values of looking to numerical. The values for Center, Right, and Left are 0, 1, 2.
- **CountIF** is used to count the number of correct predictions in comparison to ground truth.

3.2 Frameworks Outputs

3.2.1 Classification Accuracy

Accuracy simply evaluates how often the classifier predicts accurately. It is defined as the proportion of correct forecasts to total predictions.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

3.2.2 Frameworks Accuracy

This section reports the accuracy of the models in four different conditions.

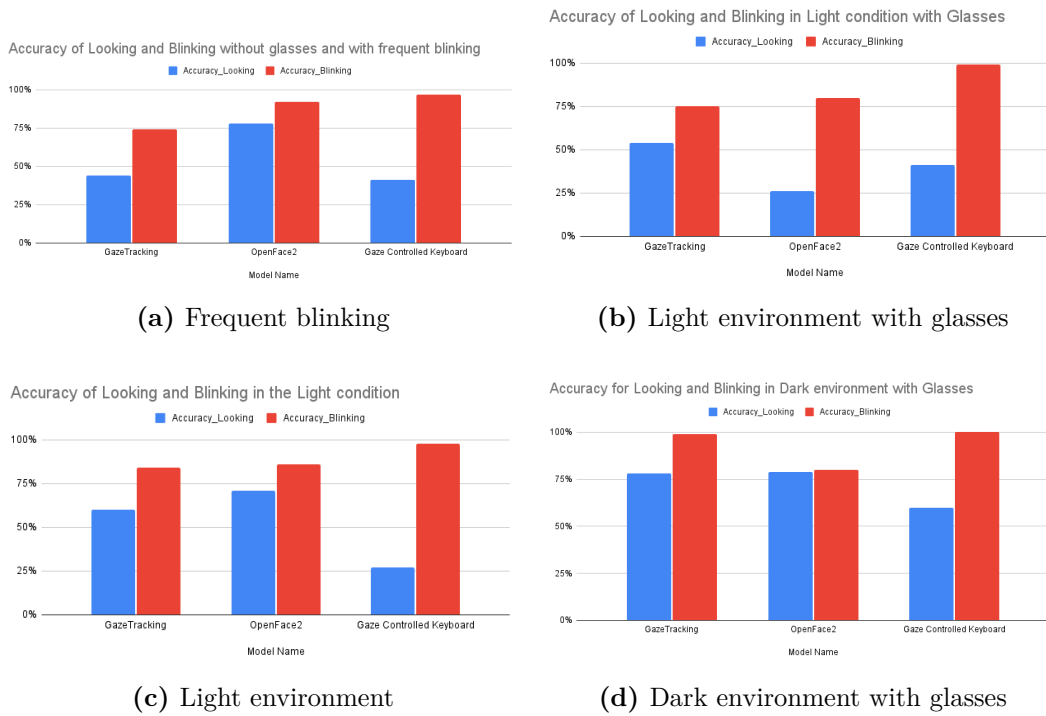


Figure 3.7: Accuracy of frameworks for gaze tracking and blinking

3.2.3 OpenFace Output Format

The OpenFace output file includes columns for gaze direction values, landmark locations in 2D, head pose, and rigid and non-rigid shape parameters; the details of each column can be found in the OpenFace GitHub repository [44]. Starting

from its output file, for our purpose, we only considered the following columns: frame, timestamp, gaze_angle_x, AU45c

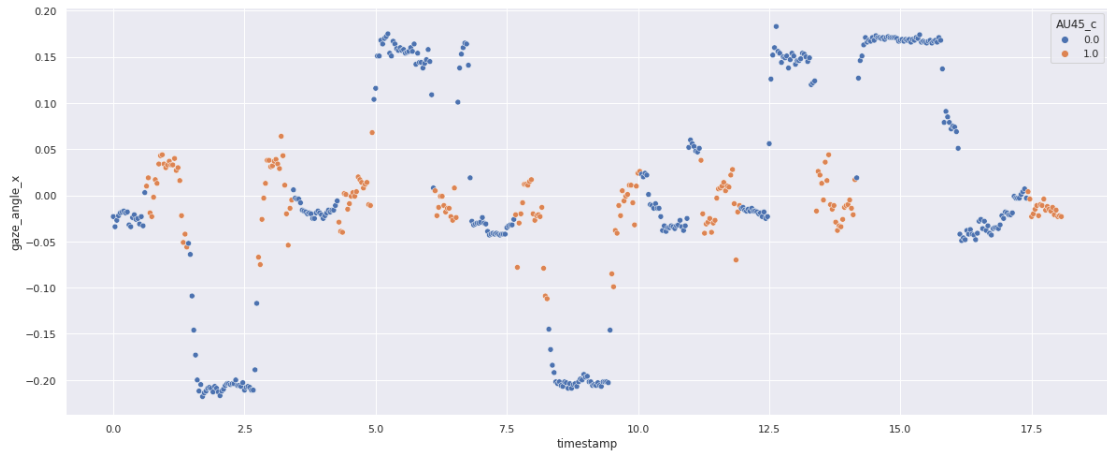


Figure 3.8: Value of gaze-angle-x column(looking right, left, center) in OpenFace for frequent blinking video

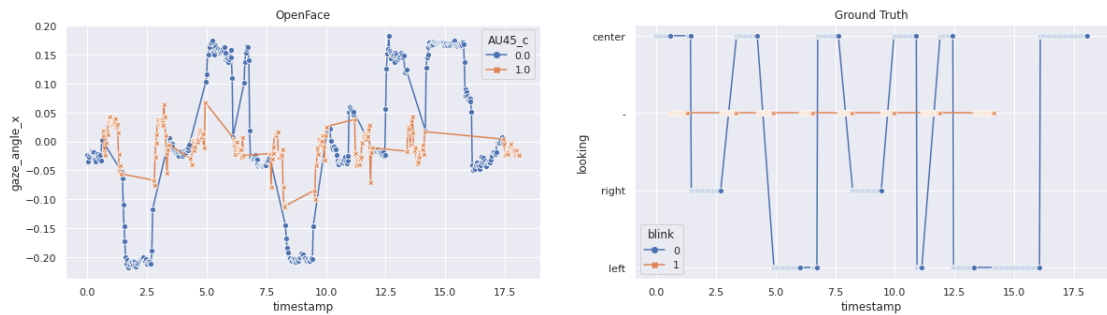


Figure 3.9: OpenFace output for gaze-angle-x versus ground truth

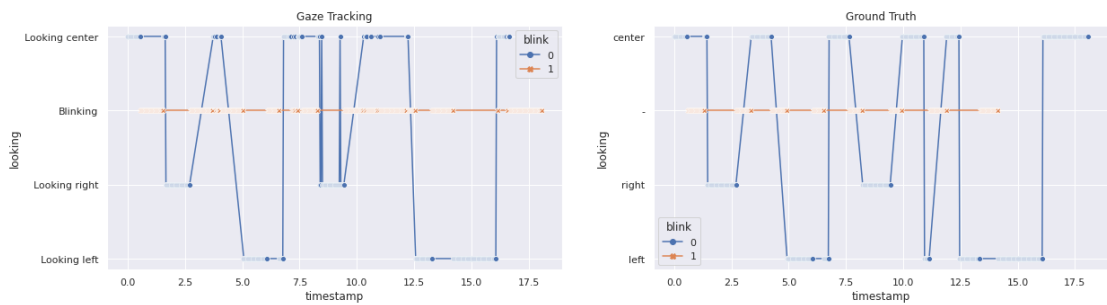


Figure 3.10: GazeTracking output for looking versus ground truth

3.2.4 Basic

- **frame** The number of the frame.
- **timestamp** The timer of video being processed in seconds (in case of sequences).
- **face_id** The face id (in case of multiple faces)
- **confidence** How confident is the tracker in current landmark detection
- **success** Is the track successful (is there a face in the frame, or did the model track it well)

3.2.5 Gaze Related

- **gaze_0_x, gaze_0_y, gaze_0_z** Direction of eye gazing (normalized) vector in world coordinates for eye 0; eye 0 is the image's leftmost eye.
- **gaze_1_x, gaze_1_y, gaze_1_z** Direction of eye gazing (normalized) vector in world coordinates for eye 1; eye 1 is the image's rightmost eye.
- **gaze_angle_x, gaze_angle_y** The average of two eyes' gaze directions in radians in world coordinates. It is a transformation of the gaze vectors into a more user-friendly format. As illustrated in Figure 3.8, when a person looks left-right, the **gaze_angle_x** changes (from positive to negative).

If the person is looking up-down this will result in change of **gaze_angle_y** (moving from negative to positive), if the person is looking straight ahead both of the angles will be close to 0 (within measurement error).

- **eye_lmk_x_0, eye_lmk_x_1, ..., eye_lmk_x55, eye_lmk_y_1, ..., eye_lmk_y_55** location of 2D eye region landmarks in pixels.
This graphic 3.11 shows the landmark index.
- **eye_lmk_X_0, eye_lmk_X_1, ..., eye_lmk_X55, eye_lmk_Y_0, ..., eye_lmk_Z_55** location of 3D eye region landmarks in millimeters.
This graphic 3.11 shows the landmark index.

3.2.6 Action Units

- The system can detect the intensity (from 0 to 5) of 17 AUs:
AU01_r, AU02_r, AU04_r, AU05_r, AU06_r, AU07_r, AU09_r, AU10_r
AU12_r, AU14_r, AU15_r, AU17_r, AU20_r, AU23_r, AU25_r, AU26_r
AU45_r

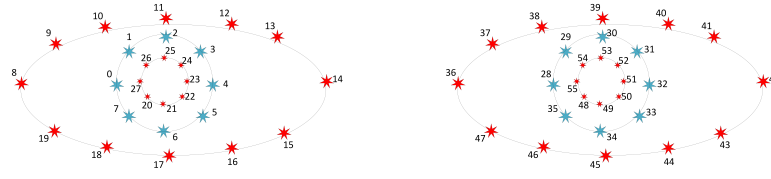


Figure 3.11: Eye landmarks. Figure from [44]

- And the presence (0 absent, 1 present) of 18 AUs: AU01_c, AU02_c, AU04_c, AU05_c, AU06_c, AU07_c, AU09_c, AU10_c, AU12_c, AU14_c, AU15_c, AU17_c, AU20_c, AU23_c, AU25_c, AU26_c, AU28_c, AU45_c

We need to value AU45_c, which detects the blink, for our requirement.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	
frame	face_id	timestamp	confidence	success	gaze_0_x	gaze_0_y	gaze_0_z	gaze_1_x	gaze_1_y	gaze_1_z	gaze_angle_x	gaze_angle_y	eye_lmk_x_0	ey
1	1	0	0.98	1	0.041444	0.175689	-0.983573	-0.08639	0.194136	-0.977163	-0.023	0.186	583.8	
2	1	0	0.033	0.98	1	0.033417	0.162411	-0.986157	-0.100333	0.177675	-0.978961	-0.034	0.171	583.6
3	1	0	0.067	0.98	1	0.041743	0.170863	-0.98441	-0.095073	0.186275	-0.977887	-0.027	0.18	583.3
4	1	0	0.1	0.98	1	0.041615	0.186618	-0.981551	-0.085374	0.203507	-0.975344	-0.022	0.197	583.4
5	1	0	0.133	0.98	1	0.045149	0.189588	-0.980825	-0.08206	0.20468	-0.975383	-0.019	0.199	583.3
6	1	0	0.167	0.98	1	0.047196	0.193732	-0.979919	-0.082247	0.206457	-0.974993	-0.018	0.202	583.2
7	1	0	0.2	0.98	1	0.045982	0.195439	-0.979637	-0.080096	0.209634	-0.974494	-0.017	0.204	583.2
8	1	0	0.233	0.98	1	0.044856	0.19781	-0.979214	-0.082543	0.204214	-0.97544	-0.019	0.203	583.1
9	1	0	0.267	0.98	1	0.048055	0.188189	-0.980956	-0.084036	0.197877	-0.976618	-0.018	0.195	583.1
10	1	0	0.3	0.98	1	0.040836	0.158867	-0.986455	-0.104481	0.1508	-0.983027	-0.032	0.156	583.3
11	1	0	0.333	0.98	1	0.034898	0.151063	-0.987908	-0.102366	0.153026	-0.982906	-0.034	0.153	583.2
12	1	0	0.367	0.98	1	0.048981	0.147446	-0.987857	-0.09681	0.148653	-0.984139	-0.024	0.149	583.1
13	1	0	0.4	0.98	1	0.050859	0.149102	-0.987513	-0.093245	0.146808	-0.98476	-0.021	0.149	583
14	1	0	0.433	0.98	1	0.042896	0.150708	-0.987647	-0.093859	0.143821	-0.985143	-0.026	0.148	582.8
15	1	0	0.467	0.98	1	0.047116	0.150062	-0.987553	-0.095582	0.14688	-0.984525	-0.025	0.149	582.8
16	1	0	0.5	0.98	1	0.038341	0.153497	-0.987405	-0.099448	0.146729	-0.984165	-0.031	0.151	582.7
17	1	0	0.533	0.98	1	0.051842	0.146567	-0.987841	-0.096475	0.148496	-0.984196	-0.023	0.149	583
18	1	0	0.567	0.98	1	0.046331	0.161958	-0.985709	-0.11194	0.162489	-0.98034	-0.033	0.164	583.3
19	1	0	0.6	0.98	1	0.084849	0.199298	-0.976259	-0.078517	0.207315	-0.975118	0.003	0.205	585.2
20	1	0	0.633	0.98	1	0.067348	0.226248	-0.971739	-0.047497	0.258688	-0.964792	0.01	0.245	586
21	1	0	0.667	0.98	1	0.089768	0.276491	-0.956815	-0.054436	0.309891	-0.949213	0.019	0.298	587.5
22	1	0	0.7	0.98	1	0.135886	0.334524	-0.932539	-0.170108	0.377337	-0.910319	-0.019	0.369	587.6
23	1	0	0.733	0.98	1	0.169425	0.42237	-0.890449	-0.210322	0.401951	-0.891179	-0.023	0.433	587.4
24	1	0	0.767	0.98	1	0.186077	0.444941	-0.876015	-0.190295	0.422592	-0.886117	-0.002	0.457	587.6
25	1	0	0.8	0.98	1	0.197723	0.436591	-0.877664	-0.16688	0.417837	-0.893064	0.017	0.45	588

Figure 3.12: CSV output of OpenFace

3.3 Selected Frameworks

We need a framework that is very good at detecting facial landmarks even when the face is not in front of the screen, as well as having a decent average accuracy for gaze tracking and blinking and running quickly and smoothly for long recorded videos. Based on our results in Section 3.2.2, we chose **OpenFace** and **Gaze tracking** as our basic frameworks for going forward and using them for integrating with an emotion detector in C++ and Python language to have our required system to detect student engagement.

The following are the reasons why we chose OpenFace:

- A more accurate facial landmark detection system.

- Accurately tracking gaze and eye.
- Complete output for our analysis or future works, in terms of head position, facial expression, and other detailed outputs of video.
- Real-time performance without the need for a GPU.

The following are the reasons why we chose GazeTracking:

- High accuracy for gaze tracking and blink detection as compared to other projects
- Simple to integrate with Emotion Detector.

Chapter 4

Emotion Detector Model

In this chapter, we will learn about Facial Expression in Section 4.1, the selection of Neural Networks for Emotion Detectors in Section 4.2, and later the possibility of doing emotion detection by AUs in Section 4.3.2, and finally in Section 4.4 the integration of the selected models from the previous chapter with the Emotion detector.

4.1 Facial Emotion Recognition

Facial Emotion Recognition (FER) is a technique that assesses emotions from various sources, including images and videos. It is part of a technology family known as *affective computing*, a multidisciplinary field of study that uses Artificial Intelligence technology to recognize and analyze human emotions and affective states.

Facial expressions are nonverbal means of communication that reveal human emotions. Decoding such emotional expressions has been a research interest for decades, not just in psychology (Ekman and Friesen 2003 [48]) but also in Human-Computer Interaction (Abdat et al. 2011 [49]).

FER can also be used with biometric identification. Its accuracy can be improved by technology that analyzes many sorts of sources such as voice, text, health data from sensors, or blood flow patterns inferred from images. Potential applications for FER include a wide range of applications, some of which are given below in groupings.

- Provision of personalized services.
- Customer behavior analysis and advertising.

- Healthcare.
- Crime detection.
- Monitoring student attention.

4.1.1 State of the Art

As humans, we are instinctively able to determine the emotions that our fellows are feeling. It is well known that facial expressions are a fundamental part of this social ability. In the 1970s, the American psychologist Paul Ekman scientifically studied this phenomenon. In 1972 [50], he published the list of the six primal emotions shared among all human groups, independently from their culture: anger, disgust, fear, happiness, sadness, and surprise. In the following years, he and other researchers added to this list other emotions.

For our purposes, we studied only eight basic emotions: the six proposed in 1972, plus contempt and neutrality. Ekman also developed the Facial Action Coding System (FACS) [51].

Facial expressions are made possible by facial muscles, which can be moved individually or in groups. These groups of muscular movements are called Action Units (AUs). As a result, facial expressions can be classified using a weighted evaluation of those AUs. These evaluations make it possible to determine facial expressions more objectively. However, the same emotion can be shown with different groups of AUs to make things even more complex. Thus, there is significant intraclass variation. If the considered facial expression is labeled by analyzing the AUs, the picture is marked as FACS encoded [52].

Furthermore, facial expressions can be posed or spontaneous: while the latter is more common to see in everyday life, the former is a more caricatural, exaggerated version of the same. Various scientists have worked on this topic over the years; hence, nowadays, many pictures of posed and spontaneous facial expressions, organized in databases, are available in the literature [52].

4.2 Choice of the Neural Network for Facial Expressions

To perform our task, we searched for some of the best neural networks explicitly created for facial expressions recognition; our choice is the model used in this paper: **Automatic Emotion Recognition for the Calibration of Autonomous Driving Functions** [52]. They chose the most suitable neural network proposed in

this paper: [53] **Physiological Inspired Deep Neural Networks for Emotion Recognition**. The chosen neural network is trained using single databases and database ensembles. The layers of this neural network are shown 4.5. In this [52] work they used the following databases:

- The Extended Cohn-Kanade (CK+) database contains 593 sequences from 123 subjects depicted in each of the eight emotional states. Each sequence begins in a neutral condition and progresses to the peak of the considered emotion. FACS codes are assigned to 327 of the 593 sequences. [54] [55]
- The Facial Expression Recognition 2013 (FER2013) Database [56] is composed of 35,887 pictures of 48×48 pixels retrieved from the Internet. Since the original labeling method has demonstrated itself erroneously in some cases, a newer set of annotations named FER+ [57] was released in 2016. It contains labels for 35,488 images since the remaining 399 do not represent human faces and adds contempt for emotion.
- The Japanese Female Facial Expression (JAFFE) collection [58] contains 213 grayscale photographs of 10 Japanese women performing posed facial expressions. 60 Japanese volunteers assessed each image on six emotional descriptors.
- The Multimedia Understanding Group (MUG) database [59] contains photos of 86 models posing six emotions: anger, disgust, fear, happiness, sadness, and surprise. The images of this database are taken inside a photographic studio, thus in controlled illumination conditions.
- The Radboud Faces Database (RaFD) [60] is a collection of photos of 67 models, posing all eight emotional states considered in this paper. Each picture was taken from five different angles simultaneously.
- SFEW 2.0 (static facial expression in the wild) collection is made up of frames from various movies representing people in seven different emotional states: anger, disgust, fear, happiness, neutrality, sadness, and surprise. They decided to use only 1694 labeled aligned images.[61]
- The FACES database has 2052 photos of 171 people. They performed each facial expression twice: anger, contempt, fear, happiness, neutrality, and sadness. The actors are further separated into three age groups.[62]

4.2.1 Neural Networks Training

For the training of the chosen neural network, they [52] chose the following databases to be able to compare the results of the implementations with those obtained by the author of neural network [53]:

- CK+
- FER2013

FEDC frequently uses the following two database ensembles:

- Ensemble 1 is made up of all the labeled photos from all of the FEDC-supported databases;
- Ensemble 2, composed of all the posed facial expressions images from the databases CK+, FACES, JAFFE, MUG, and RaFD.

Table 4.1 indicates the number of pictures for each emotion in the chosen databases.

Emotion	Ensemble 1	Ensemble 2	CK+	FER2013
Anger	4328	981	45	4953 3111 ^a
Contempt	788	572	18	0 216 ^a
Disgust	1340	1022	59	547 248 ^a
Fear	1887	950	25	5121 819 ^a
Happiness	10,676	1089	69	8989 9355 ^a
Neutrality	14,196	1070	123	6198 12,906 ^a
Sadness	5524	953	28	6077 4371 ^a
Surprise	5254	656	83	4002 4462 ^a

Table 4.1: Available pictures for each emotion in the chosen databases [52]
^aFER+ annotations

4.3 Facial Action Coding System

The Facial Action Coding System (FACS) categorizes human facial actions based on how they appear on the face. FACS can code practically any physically possible facial expression, breaking it down into the individual Action Units (AU) that formed the emotion. It is a commonly used criterion for objectively describing facial expressions.[44] You can learn more about FACS and AUs here [63]



Figure 4.1: AUs. Figure from[63]

4.3.1 Intensity and Presence of AUs in OpenFace

As we mentioned in this Section 2.8.4, the OpenFace can detect the AUs; it can be described in two ways:

- Presence: If AU is visible in the face (for example, AU01-c).
- Intensity: On a 5-point scale, how intense is the AU (from minimum to maximum).

OpenFace provides both of these scores. For the presence of AU01, the column AU01-c in the output file would encode zero as not present and one as present. The intensity of AU01 would be represented in the output file by the column AU01-r, which would have values ranging from 0 (not present), 1 (present at minimal intensity), and 5 (present at highest intensity), with continuous values in between. Because the intensity and presence predictors were trained independently and on slightly different datasets, their predictions may not always be equivalent. For more information, check out the datasets used to train the model.[44]

4.3.2 Can AUs be used to detect facial emotions?

One of our concerns was whether we could map the AUs to certain facial emotions. Because OpenFace provides this output for each frame. OpenFace does not predict basic emotions from AUs since the mapping is more complex than the Wikipedia website says [63]. Emotional expression is very contextual, and the same expression might mean many things depending on the situation, culture, age, and other factors of the person displaying it.

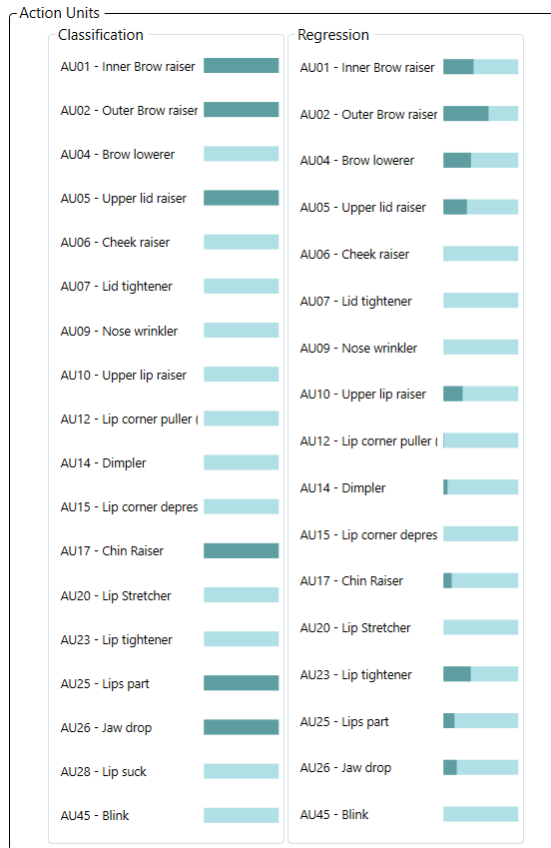


Figure 4.2: Presence and intensity of AUs Detected by OpenFace

4.4 Integrated System

For running the selected neural network for our purpose, we used a utility code: **Real-time Facial Emotion Detection using deep learning in Python** [64]. The project used OpenCV and Tensorflow libraries. We integrate this project with the GazeTracking framework that we discussed before in Section 2.8.9; a screenshot of this application is 4.3, and it works according to this 4.4 and the below steps:

- The Haar cascade approach is used to recognize faces in each frame of the webcam feed.
- The portion of the image containing the face is scaled to 48x48 and fed into the CNN.
- The network generates a list of softmax scores for each of the eight emotion classes.

- The emotion with the highest score is shown on the screen.
- Each frame's emotion is recorded in a CSV file.

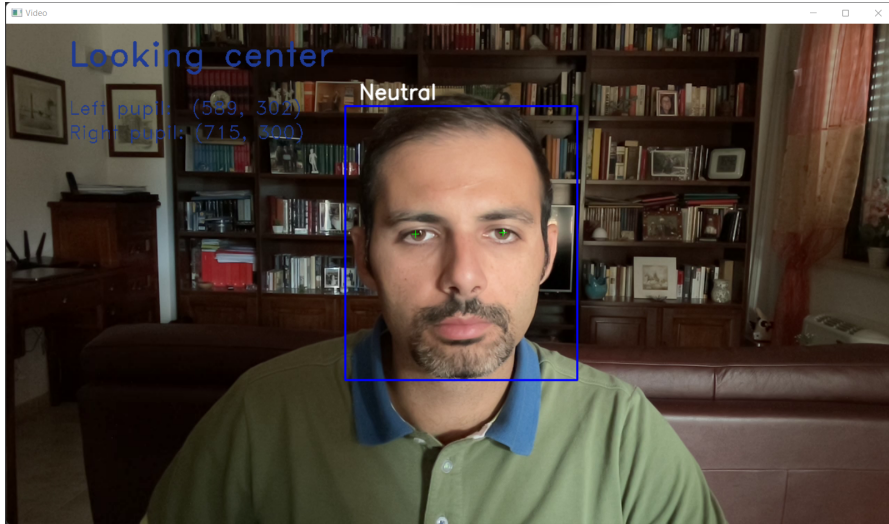


Figure 4.3: The emotion and gaze direction of the subject are displayed by the GazeTracking application.

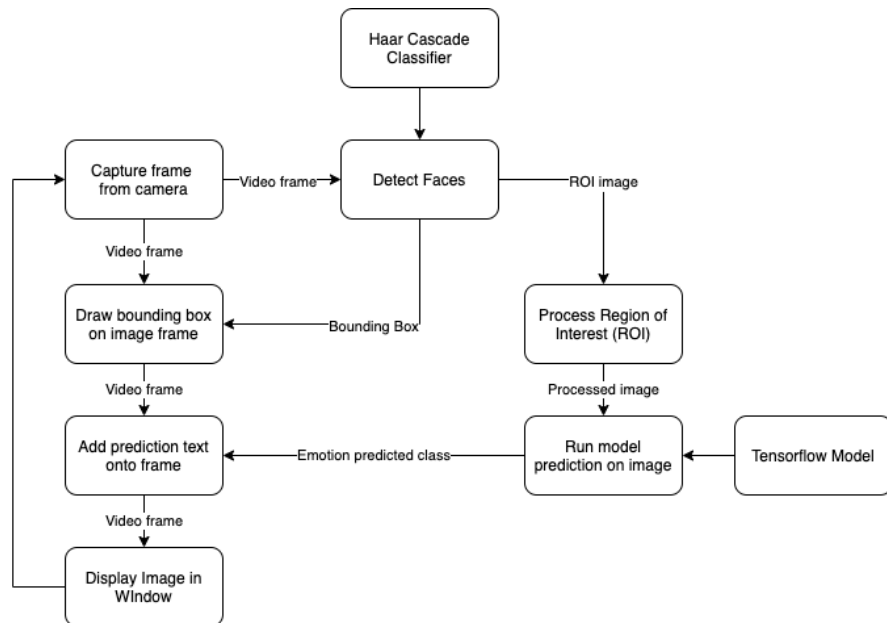


Figure 4.4: Structure of the Emotion Detector code. Figure from [64]

4.4.1 Result of Emotion Detector

These graphs 4.6 show the results of the emotion detector on the example videos. **Neutrality** is the most prominent emotion because the subject does not have any interactions with anyone.

4.4.2 Integrating Emotion Detector model with OpenFace

As said in this section 2.8.4, OpenFace can detect AUs; nevertheless, it is difficult to reach a correct label for facial emotion based on those AUs; hence we add the feature of real-time emotion detection to OpenFace. Since OpenFace is written in C++, we require C++ code to infer the trained model for detecting emotions; with the help of this utility code [65], We added this feature to only detect facial emotions in live webcam and save the results in a CSV file. Figure 4.7 shows it.

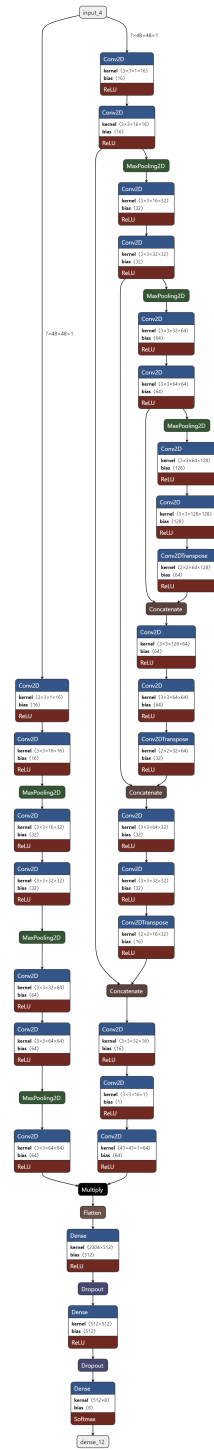


Figure 4.5: Visualization of the Tensorflow model for detecting facial expression by the Netron app

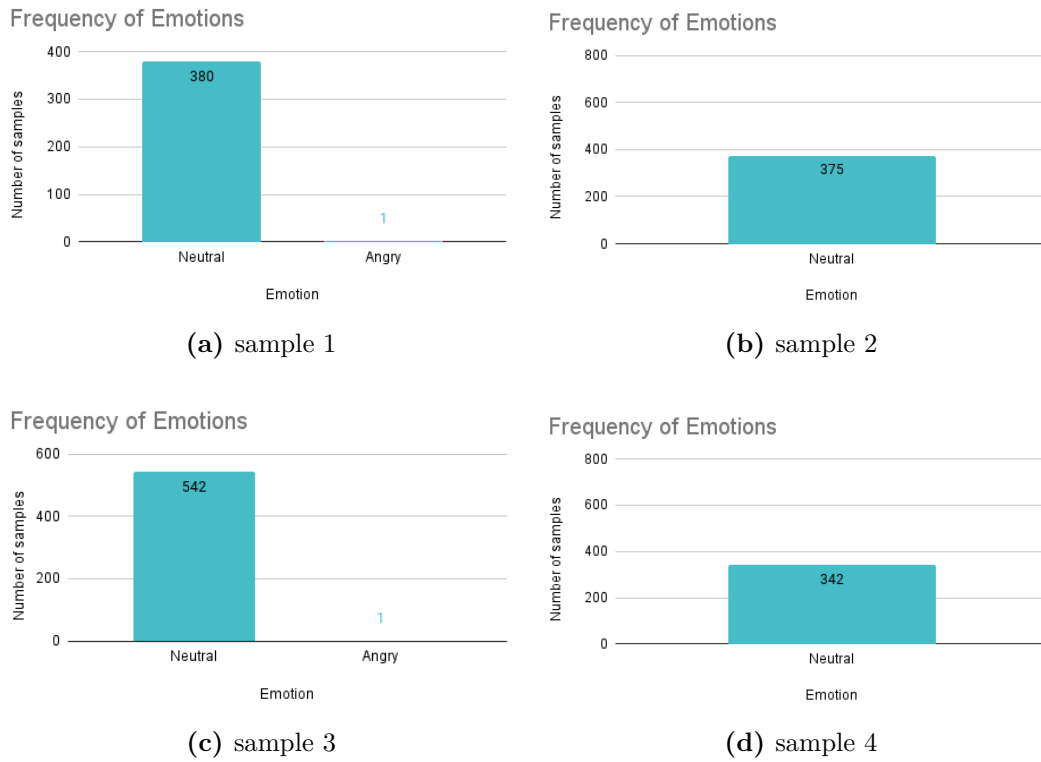


Figure 4.6: Frequency of Emotions in four sample recorded videos

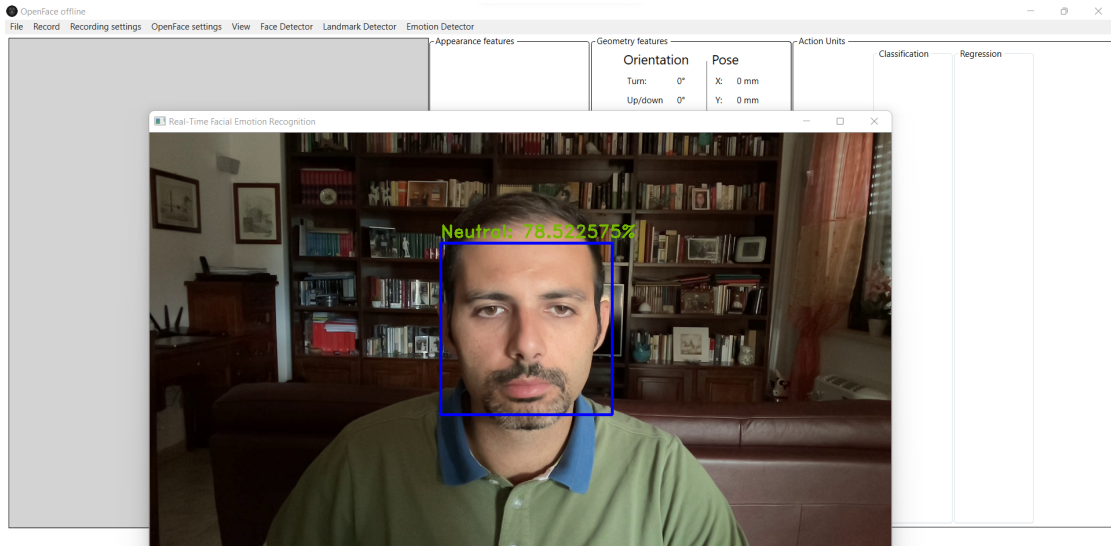


Figure 4.7: Live webcam facial emotion detector in OpenFace

Chapter 5

Detecting Engagement

Our experiment in this chapter is in the subsequent work for improving this article: **A Novel Redundant Validation IoT System for Affective Learning Based on Facial Expressions and Biological Signals** [24], by adding gaze tracking and blinking features. Starting with the solution described in the previous paper, the goal is to determine whether or not the gaze value and blinking are effective for determining the level of student attention in remote lecture. This figure 5.1 depicts the workflow of their proposed analysis approach.

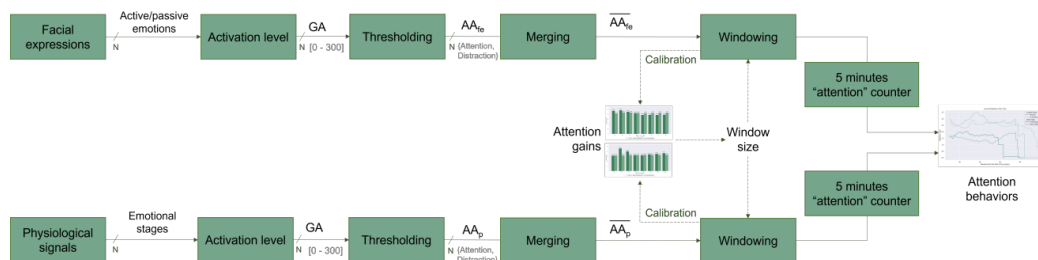


Figure 5.1: The workflow of the studied approach. Figure from [24]

5.1 Engagement in Remote Lectures

The activity of teaching necessitates constantly examining oneself and guessing independently whether or not the topics are being explained properly. Such abilities are typically improved by observation of classroom reactions to various approaches and lecture content. In small classrooms, this task may be easy, but in large classrooms with lots of learners, it may be challenging. Large classrooms can also

make it difficult for students to interact with teachers and for lessons to be seen and heard [66].

In our experiment, facial expression recognition and eye state (gaze tracking and blinking) are employed as primary sources, while the physiological data analysis, since it requires students to wear a smartwatch, is used as a validation tool when available. We will go over recorded videos in detail in the Experiment section 5.3.

5.2 Improvements towards an Attention Detection Algorithm

This section explains the steps done toward a physiologically based Attention Detection algorithm. As in the proposal of the paper [24], physiological data were collected using a set of commercial smartwatches (Garmin Venu Sq) that send data through Bluetooth Low Energy (BLE) to a set of smartphones at a sampling rate of 1 Hz. The information gathered is kept in a text file. The acquired datasets include HR and HRV. The algorithm was created entirely in MATLAB; before executing it, each patient goes through a two-level calibration phase focusing on estimating thresholds for emotional stage detection:

- Based on their information, which are age, gender, weight, and height.
- From their initial condition estimation.

The algorithm may undertake a behavioral analysis in real-time during the lecture. This includes observing the HR and HRV for N samples, which is defined as Window Size in the following (WS).

Only a few of the emotional stages described are used from this observation window: SDNN, RMSS, and SDD. It is possible to generate an output every second by sliding the window with a set initial delay.

It is possible to raise the Grade of Attention by comparing the punctual value of the three parameters with the one obtained during the calibration phase (GA). Similarly to the classification of attention levels, GA is compared to a Threshold (T_p), which is computed as:

$$T_p = \frac{3}{2} \times WS$$

Where 3 is the total number of emotional stages calculated. Thus, two levels from physiological reactions (AAp) were acquired and stored in an Attention Array. In terms of neural network classification, if the grade of attention is greater than T_p , the window is labeled *Attention* otherwise *Distraction*.

5.3 Experiment

For this section, we have three recorded videos of students who attended a pre-recorded lecture for 50 minutes, and their faces are recorded during the whole lecture. They were also given smartwatches to track their physiological reactions. Furthermore, because there was no professor present, an application was utilized to randomly question students whether they were focusing on the lecture or not. The source code of this application is available for interested readers on GitHub under the MIT license [66].

5.3.1 Application to Measure the Response Time

The application, called Reaction Time Tool, was built with Microsoft.NET 6 framework and C sharp programming language. The Windows Presentation Foundation (WPF) environment was used to create the Human-Machine Interface (HMI).

The software consists of two primary windows: the first appears at program launch and asks volunteers where they want to keep the log file, and the second displays a blinking hourglass to inform the user that the application is functioning properly. A message window appears at predefined moments to the volunteers, asking if they are paying attention to the lecture.

The user can respond *Yes* or *No* depending on their level of attention. This window cannot be closed without an answer. When the message window appears, the application makes a sound to get the user's attention. This sound was designed to be distinct from the sounds frequently made by operating system message windows. When the window is displayed, the application records the timestamp of the event in the log file, while, when it is closed, a second event is saved alongside the answer provided by the volunteers. Furthermore, to make data analysis easier, the application computes the time between the display of the window and the user response. This application, of course, does not run on a real-time system, but on a standard personal computer. The purposes of this application are twofold:

1. It asks the students if they are paying attention to the lecture.
2. It records the reaction time of the student [24].

5.3.2 Our choice for Software

During our experiment, we discovered that OpenFace has superior performance for processing videos and generating output, thus we conducted the complete experiment with this software.

The recorded videos are passed into OpenFaceOffline, where the facial emotions are

detected using the Emotion Detector static library that we integrated to OpenFace; the results are saved as a CSV file.

5.3.3 Standard Deviation

The Standard Deviation is commonly used to compute the spread and variation of a set of data values. It is related to variance in that it offers the measure of deviation, whereas variance provides the squared value. Its formula is the following:

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$$

where $x_1, x_2, x_3, \dots, x_N$ are observed values in sample data, \bar{x} is the mean value of observations and N is the number of sample observations.

A low value shows that the data in a set are less split apart from their mean average values, whereas a high value indicates that the data in a set are more spread out from their mean average values. The standard deviation, unlike variance, is represented in the same units as the data. The Statistics package in Python includes a function called `stdev()` that can be used to compute it from a subset of data rather than the entire population [67].

5.4 Experimental Results

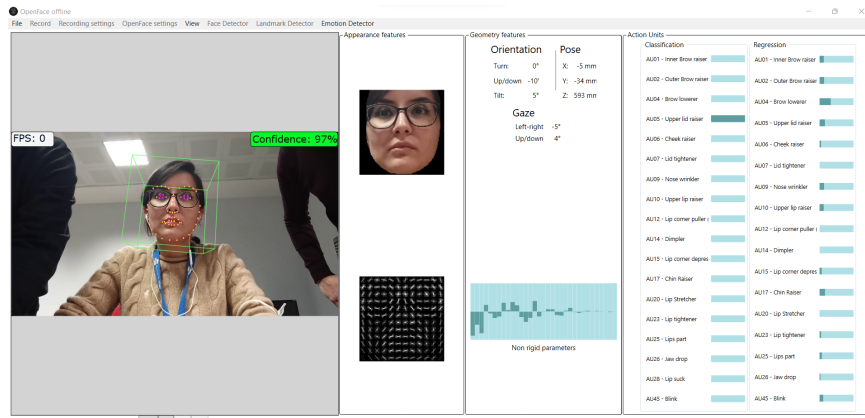
5.4.1 Gaze Tracking

We computed the standard deviation of the `gaze-angle-x` for rows with timestamps ranging from 10 to 20 minutes. As previously stated in Section 3.2.3, the `gaze-angle-x` value specifies whether the subject is looking to the right, left, or center of the screen.

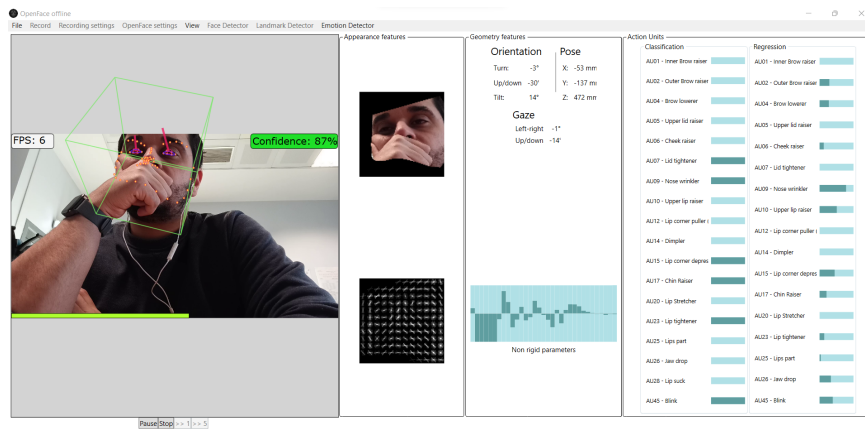
For the first run of the standard deviation function, receive the current row value of gaze with the next batch rows (in our code, we set the batch size to 3000 rows), and the result will be placed in another column. These operations will be repeated for each subsequent row until the entire set is reached.

The ground truth from biological data is illustrated in Figure 5.10b, shows the level of attention where 4 denotes a high level of focus and 0 implies no focus. As seen in Figure 5.10b, the sum of standard deviation of three samples is consistent with the ground truth.

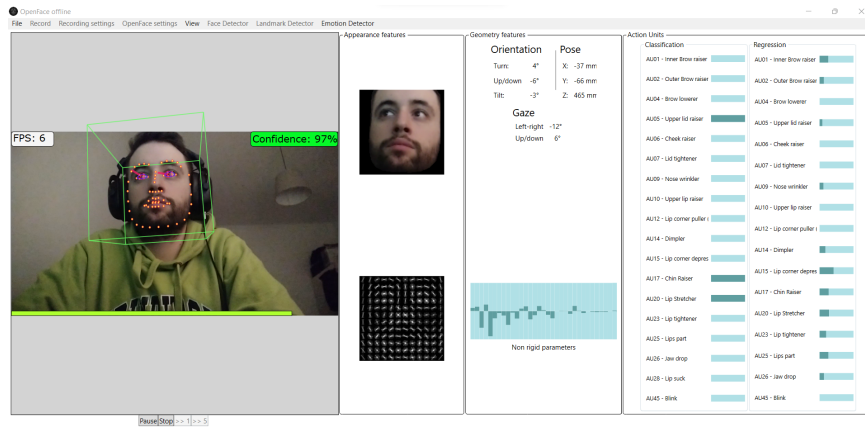
Detecting Engagement



(a) Sample 1

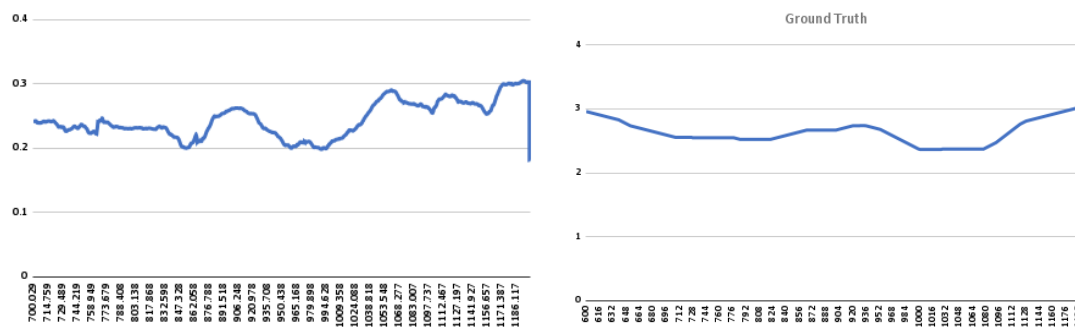


(b) Sample 2



(c) Sample 3

Figure 5.2: Running the OpenFace on the three recorded videos



(a) Sum of standard deviation of three student’s gaze during 10 minutes following the remote lecture

(b) The Ground truth of engagement

Figure 5.3: From the charts displayed, it is possible to notice that (a) is aligned with the ground truth of engagement (b)

As shown in Figure 5.3a, the total of standard deviations for three students’ gaze values are aligned with the right plot, which depicts the ground truth, confirming that gaze value is a significant component in detecting the level of engagement.

5.4.2 Facial Emotion

After running the emotion detector model on the three samples, the most significant facial emotion in all of them is **Neutral**. These charts 5.8a demonstrate the frequency of emotions for 10 minutes of recording samples.

5.4.3 Data Processing Tools

We used Google Colaboratory environment to import the CSV files as data frames as we have huge files with many rows. We used Matplotlib and Pandas libraries of Python to plot the results.

```
[ ] from pathlib import PureWindowsPath
import statistics as stats

def interval_sdv(v, i, b):
    a = len(v)
    try:
        sdv = stats.stdev(v[i : i + b])

        return sdv
    except Exception as Exc:
        print(str(Exc))

[ ] v = new_df3['gaze_angle_x']
b = 3000 #the batch that goes in each iteration

a = len(v)
sdv_array = []
count = 0
i = 0
while i < a - b:
    sdv = interval_sdv(v, i, b)
    ### write sdv in new column, index i
    sdv_array.append(sdv)
    i += 1
    count += 1

print(count)
print(sdv_array)
```

Figure 5.4: Python code for computing the standard deviation on an input column

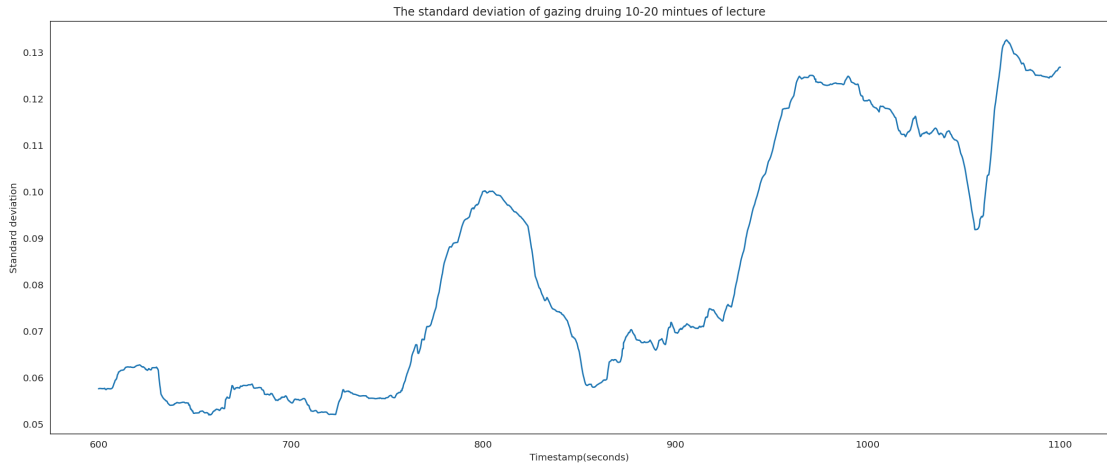


Figure 5.5: standard deviation for gaze value in sample video 1

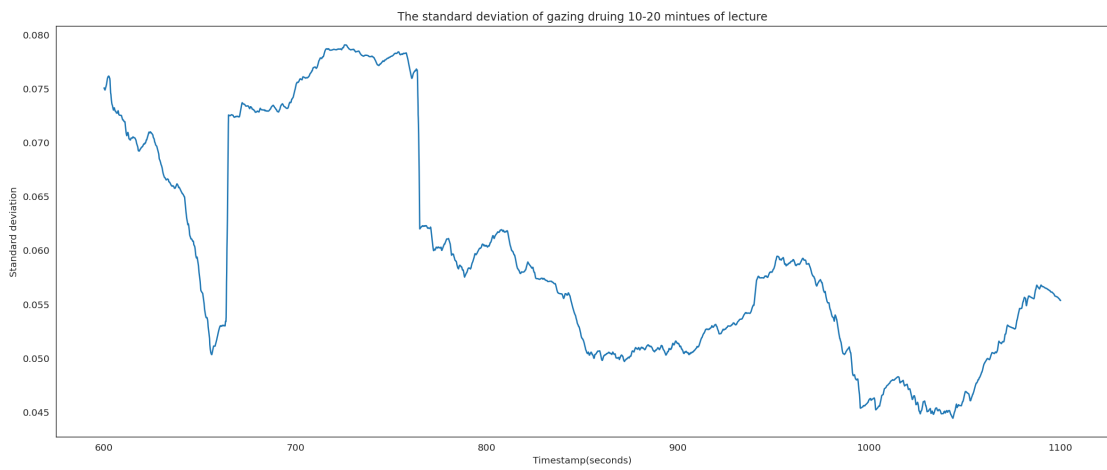


Figure 5.6: standard deviation for gaze value in sample video 2

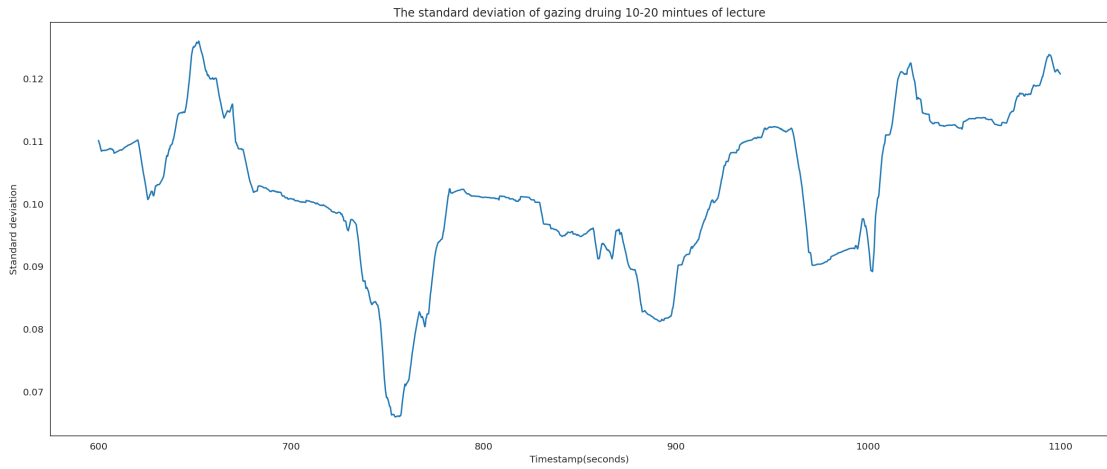
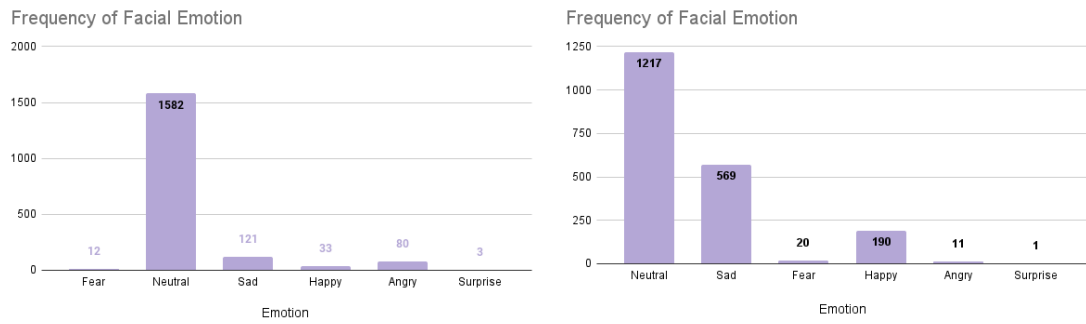
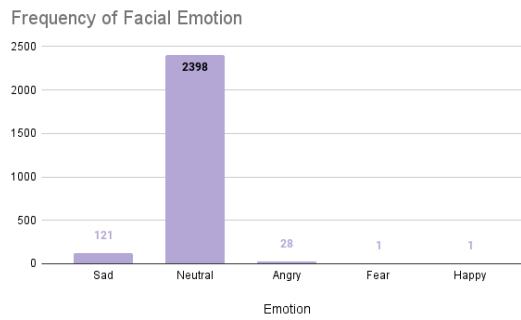


Figure 5.7: standard deviation for gaze value in sample video 3



(a) Sample 1

(b) Sample 2



(c) Sample 3

Figure 5.8: Frequency of facial emotions in three samples; The dominant emotion is Neutral.

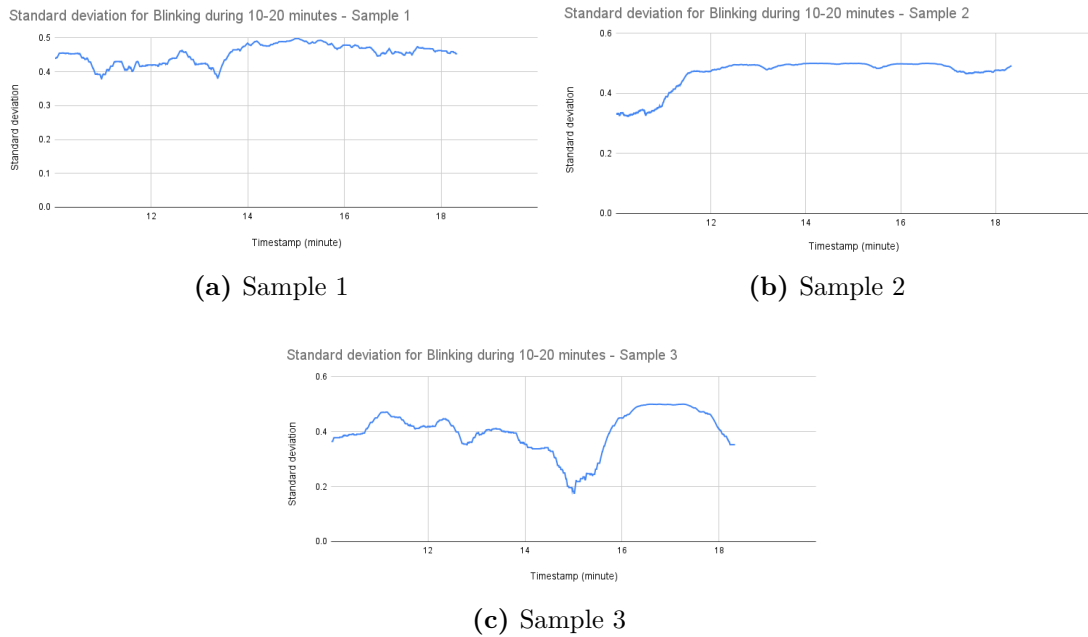


Figure 5.9: Standard deviation of Blinking for three samples

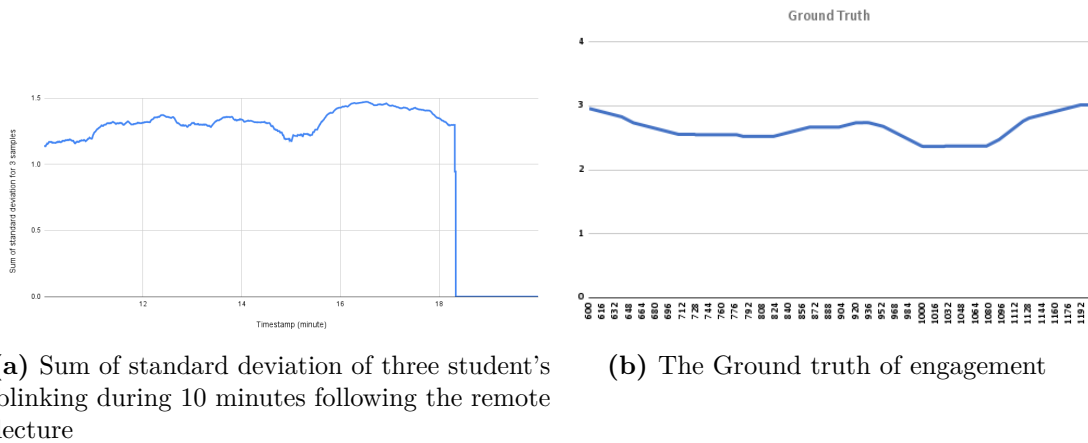


Figure 5.10: The sum of standard deviations for blinking in comparison to the ground truth

Chapter 6

Conclusions

6.1 Conclusions

This research aims to improve this article [24] by incorporating a gaze tracking function and eye blink detection to determine whether the student attending the remote lecture is focused or distracted. In this work, we used the ground truths provided in the aforementioned paper and, by analyzing the gaze value of three students for 10 minutes, we discovered that the gaze value is compatible with the ground truth, and thus we can confirm that it is an important factor in detecting the level of engagement.

In our work, we evaluate different applications for tracking the eye and detecting the blink, as well as detecting the facial expression, which provides us with the output for analysis. Since most facial analysis tools are not free and open source, else they could be free but inaccurate and lack the performance we require for our video processing. We chose the **OpenFace** software that meets our needs in this study.

6.2 Future Works

A lot of work has been done throughout the thesis, but it can be improved further. Following are some suggestions for future works.

- More accurate face detectors could be used. Furthermore, the neural network responsible for recognizing facial expressions could be improved by studying new models, using state-of-the-art datasets, and doing a thorough search for hyperparameters.
- Examining other features that OpenFace provides as output, such as head pose and looking up/down.

- Finally, student engagement detection can be improved by using other features or data sources such as activity recognition and head pose estimation.

Bibliography

- [1] Peijin Chen. «Research on sharing economy and e-learning in the era of “Internet plus”». In: *2018 2nd International Conference on Education Science and Economic Management (ICESEM 2018)*. Atlantis Press. 2018, pp. 751–754 (cit. on p. 1).
- [2] Bin Zhu, Xinjie Lan, Xin Guo, Kenneth E. Barner, and Charles Boncelet. «Multi-Rate Attention Based GRU Model for Engagement Prediction». In: *Proceedings of the 2020 International Conference on Multimodal Interaction*. ICMI '20. Virtual Event, Netherlands: Association for Computing Machinery, 2020, pp. 841–848. ISBN: 9781450375818. DOI: 10.1145/3382507.3417965. URL: <https://doi.org/10.1145/3382507.3417965> (cit. on p. 1).
- [3] Abhay Gupta, Arjun D’Cunha, Kamal Awasthi, and Vineeth Balasubramanian. «Daisee: Towards user engagement recognition in the wild». In: *arXiv preprint arXiv:1609.01885* (2016) (cit. on p. 1).
- [4] Jianming Wu, Bo Yang, Yanan Wang, and Gen Hattori. «Advanced multi-instance learning method with multi-features engineering and conservative optimization for engagement intensity prediction». In: *Proceedings of the 2020 International Conference on Multimodal Interaction*. 2020, pp. 777–783 (cit. on p. 1).
- [5] James Bowers and Poonam Kumar. «Students’ perceptions of teaching and social presence: A comparative analysis of face-to-face and online learning environments». In: *International Journal of Web-Based Learning and Teaching Technologies (IJWLTT)* 10.1 (2015), pp. 27–44 (cit. on p. 1).
- [6] Javier Hernandez-Ortega, Roberto Daza, Aythami Morales, Julian Fierrez, and Javier Ortega-Garcia. «edBB: Biometrics and behavior for assessing remote education». In: *arXiv preprint arXiv:1912.04786* (2019) (cit. on p. 1).
- [7] Roberto Daza, Aythami Morales, Julian Fierrez, and Ruben Tolosana. «MEBAL: A multimodal database for eye blink detection and attention level estimation». In: *Companion Publication of the 2020 International Conference on Multimodal Interaction*. 2020, pp. 32–36 (cit. on p. 1).

- [8] Javier Hernandez-Ortega, Julian Fierrez, Aythami Morales, and David Diaz. «A comparative evaluation of heart rate estimation methods using face videos». In: *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE. 2020, pp. 1438–1443 (cit. on p. 1).
- [9] Liping Shen, Minjuan Wang, and Ruimin Shen. «Affective e-learning: Using “emotional” data to improve learning in pervasive learning environment». In: *Journal of Educational Technology & Society* 12.2 (2009), pp. 176–189 (cit. on p. 1).
- [10] Stylianos Asteriadis, Paraskevi Tzouveli, Kostas Karpouzis, and Stefanos Kollias. «Estimation of behavioral user state based on eye gaze and head pose—application in an e-learning environment». In: *Multimedia Tools and Applications* 41.3 (2009), pp. 469–493 (cit. on p. 1).
- [11] Shimeng Peng, Lujie Chen, Chufan Gao, and Richard Jiarui 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. 10. 2020, pp. 13895–13896 (cit. on p. 1).
- [12] Michele Nappi, Stefano Ricciardi, Massimo Tistarelli, et al. «Context awareness in biometric systems and methods: State of the art and future scenarios». In: *IMAGE AND VISION COMPUTING* 76 (2018), pp. 27–36 (cit. on p. 1).
- [13] Julian Fierrez-Aguilar, Daniel Garcia-Romero, Javier Ortega-Garcia, and Joaquin Gonzalez-Rodriguez. «Adapted user-dependent multimodal biometric authentication exploiting general information». In: *Pattern Recognition Letters* 26.16 (2005), pp. 2628–2639 (cit. on p. 2).
- [14] Janice Bagley and Leon Manelis. «Effect of awareness on an indicator of cognitive load». In: *Perceptual and Motor Skills* 49.2 (1979), pp. 591–594 (cit. on p. 2).
- [15] Morris K Holland and Gerald Tarlow. «Blinking and mental load». In: *Psychological Reports* 31.1 (1972), pp. 119–127 (cit. on p. 2).
- [16] Samantha Mann, Aldert Vrij, and Ray Bull. «Suspects, lies, and videotape: An analysis of authentic high-stake liars». In: *Law and human behavior* 26.3 (2002), pp. 365–376 (cit. on p. 2).
- [17] Essa R Anas, Pedro Henriquez, and Bogdan J Matuszewski. «Online eye status detection in the wild with convolutional neural networks». In: *International Conference on Computer Vision Theory and Applications*. Vol. 7. SciTePress. 2017, pp. 88–95 (cit. on p. 2).

- [18] Beatriz Remeseiro, Alba Fernández, and Madalena Lira. «Automatic eye blink detection using consumer web cameras». In: *International Work-Conference on Artificial Neural Networks*. Springer. 2015, pp. 103–114 (cit. on p. 2).
- [19] Tereza Soukupova and Jan Cech. «Eye blink detection using facial landmarks». In: *21st computer vision winter workshop, Rimske Toplice, Slovenia*. 2016 (cit. on p. 2).
- [20] Guilei Hu, Yang Xiao, Zhiguo Cao, Lubin Meng, Zhiwen Fang, Joey Tianyi Zhou, and Junsong Yuan. «Towards real-time eyeblink detection in the wild: Dataset, theory and practices». In: *IEEE Transactions on Information Forensics and Security* 15 (2019), pp. 2194–2208 (cit. on p. 2).
- [21] Minsu Jang et al. «Building an automated engagement recognizer based on video analysis». In: *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*. 2014, pp. 182–183 (cit. on p. 2).
- [22] Hamed Monkaresi, Nigel Bosch, Rafael A Calvo, and Sidney K D’Mello. «Automated detection of engagement using video-based estimation of facial expressions and heart rate». In: *IEEE Transactions on Affective Computing* 8.1 (2016), pp. 15–28 (cit. on p. 2).
- [23] Jacob Whitehill, ZewelANJI Serpell, Yi-Ching Lin, Aysha Foster, and Javier R Movellan. «The faces of engagement: Automatic recognition of student engagement from facial expressions». In: *IEEE Transactions on Affective Computing* 5.1 (2014), pp. 86–98 (cit. on p. 2).
- [24] Antonio Costantino Marceddu et al. «A Novel Redundant Validation IoT System for Affective Learning Based on Facial Expressions and Biological Signals». In: *Sensors* 22.7 (2022), p. 2773 (cit. on pp. 3, 45–47, 55).
- [25] Varun. *What is Face Detection? – The Ultimate Guide for 2022*. <https://learnopencv.com/what-is-face-detection-the-ultimate-guide> (cit. on pp. 6–9, 11, 12).
- [26] Paul Viola and Michael Jones. «Rapid object detection using a boosted cascade of simple features». In: *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*. Vol. 1. Ieee. 2001, pp. I–I (cit. on p. 6).
- [27] OpenCv. *Cascade Classifier*. https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html (cit. on pp. 6, 7).
- [28] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. «Ssd: Single shot multibox detector». In: *European conference on computer vision*. Springer. 2016, pp. 21–37 (cit. on p. 7).

- [29] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. «Joint face detection and alignment using multitask cascaded convolutional networks». In: *IEEE signal processing letters* 23.10 (2016), pp. 1499–1503 (cit. on p. 8).
- [30] Jian Li, Yabiao Wang, Changan Wang, Ying Tai, Jianjun Qian, Jian Yang, Chengjie Wang, Jilin Li, and Feiyue Huang. «DSFD: dual shot face detector». In: (2019), pp. 5060–5069 (cit. on pp. 8, 9).
- [31] Jiankang Deng, Jia Guo, Evangelos Ververas, Irene Kotsia, and Stefanos Zafeiriou. «Retinaface: Single-shot multi-level face localisation in the wild». In: (2020), pp. 5203–5212 (cit. on p. 9).
- [32] Valentin Bazarevsky, Yury Kartynnik, Andrey Vakunov, Karthik Raveendran, and Matthias Grundmann. «Blazeface: Sub-millisecond neural face detection on mobile gpu». In: *arXiv preprint arXiv:1907.05047* (2019) (cit. on p. 10).
- [33] MediaPipe. *MediaPipe Face Detection*. https://google.github.io/mediapipe/solutions/face_detection.html (cit. on p. 10).
- [34] Yuantao Feng, Shiqi Yu, Hanyang Peng, Yan-Ran Li, and Jianguo Zhang. «Detect Faces Efficiently: A Survey and Evaluations». In: *IEEE Transactions on Biometrics, Behavior, and Identity Science* 4.1 (2021), pp. 1–18 (cit. on p. 10).
- [35] Aqeel Anwar. *What is Average Precision in Object Detection & Localization Algorithms and how to calculate it?* <https://towardsdatascience.com/what-is-average-precision-in-object-detection-localization-algorithms-and-how-to-calculate-it-3f330efe697b> (cit. on p. 11).
- [36] Dlib. *Dlib*. <http://dlib.net> (cit. on p. 13).
- [37] Vahid Kazemi and Josephine Sullivan. «One millisecond face alignment with an ensemble of regression trees». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 1867–1874 (cit. on p. 13).
- [38] Jan Cech and Tereza Soukupova. «Real-time eye blink detection using facial landmarks». In: *Cent. Mach. Perception, Dep. Cybern. Fac. Electr. Eng. Czech Tech. Univ. Prague* (2016), pp. 1–8 (cit. on pp. 14, 15).
- [39] Tadas Baltrusaitis, Amir Zadeh, Yao Chong Lim, and Louis-Philippe Morency. «Openface 2.0: Facial behavior analysis toolkit». In: *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*. IEEE. 2018, pp. 59–66 (cit. on pp. 16–19).
- [40] Amir Zadeh, Yao Chong Lim, Tadas Baltrusaitis, and Louis-Philippe Morency. «Convolutional experts constrained local model for 3d facial landmark detection». In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2017, pp. 2519–2528 (cit. on p. 16).

- [41] Tadas Baltrušaitis, Peter Robinson, and Louis-Philippe Morency. «Constrained local neural fields for robust facial landmark detection in the wild». In: *Proceedings of the IEEE international conference on computer vision workshops*. 2013, pp. 354–361 (cit. on p. 17).
- [42] Erroll Wood, Tadas Baltrušaitis, Xucong Zhang, Yusuke Sugano, Peter Robinson, and Andreas Bulling. «Rendering of eyes for eye-shape registration and gaze estimation». In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 3756–3764 (cit. on p. 17).
- [43] Tadas Baltrušaitis, Marwa Mahmoud, and Peter Robinson. «Cross-dataset learning and person-specific normalisation for automatic action unit detection». In: *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*. Vol. 6. IEEE. 2015, pp. 1–6 (cit. on p. 18).
- [44] Tadas Baltrušaitis. *OpenFaceGithub*. <https://github.com/TadasBaltrušaitis/OpenFace> (cit. on pp. 18, 30, 33, 38, 39).
- [45] Sergio Canu. *GazeControlledKeyboard*. <https://pysource.com/category/tutorials/gaze-controlled-keyboard/> (cit. on pp. 22, 23).
- [46] Asadullah Dal. *Eyes Position Estimator Mediapipe*. <https://github.com/Asadullah-Dal17/Eyes-Position-Estimator-Mediapipe> (cit. on p. 23).
- [47] Antoine Lamé. *GazeTracking*. <https://github.com/antoinelame/GazeTracking> (cit. on pp. 24–26).
- [48] Paul Ekman and Wallace V Friesen. *Unmasking the face: A guide to recognizing emotions from facial clues*. Vol. 10. Ishk, 2003 (cit. on p. 35).
- [49] Faiza Abdat, Choubeila Maaoui, and Alain Pruski. «Human-computer interaction using emotion recognition from facial expression». In: *2011 UKSim 5th European Symposium on Computer Modeling and Simulation*. IEEE. 2011, pp. 196–201 (cit. on p. 35).
- [50] Paul Ekman. «Basic emotions». In: *Handbook of cognition and emotion* 98.45-60 (1999), p. 16 (cit. on p. 36).
- [51] Paul Ekman and Wallace V Friesen. «Facial action coding system». In: *Environmental Psychology & Nonverbal Behavior* (1978) (cit. on p. 36).
- [52] Jacopo Sini, Antonio Costantino Marceddu, and Massimo Violante. «Automatic Emotion Recognition for the Calibration of Autonomous Driving Functions». In: *Electronics* 9.3 (2020). ISSN: 2079-9292. DOI: 10.3390/electronics9030518. URL: <https://www.mdpi.com/2079-9292/9/3/518> (cit. on pp. 36–38).

- [53] Pedro M Ferreira, Filipe Marques, Jaime S Cardoso, and Ana Rebelo. «Physiological inspired deep neural networks for emotion recognition». In: *IEEE Access* 6 (2018), pp. 53930–53943 (cit. on pp. 37, 38).
- [54] Takeo Kanade, Jeffrey F Cohn, and Yingli Tian. «Comprehensive database for facial expression analysis». In: *Proceedings fourth IEEE international conference on automatic face and gesture recognition (cat. No. PR00580)*. IEEE. 2000, pp. 46–53 (cit. on p. 37).
- [55] Patrick Lucey, Jeffrey F Cohn, Takeo Kanade, Jason Saragih, Zara Ambadar, and Iain Matthews. «The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression». In: *2010 IEEE computer society conference on computer vision and pattern recognition-workshops*. IEEE. 2010, pp. 94–101 (cit. on p. 37).
- [56] Ian J. Goodfellow et al. «Challenges in representation learning: A report on three machine learning contests». In: *Neural Networks* 64 (2015). Special Issue on “Deep Learning of Representations”, pp. 59–63. ISSN: 0893-6080. DOI: <https://doi.org/10.1016/j.neunet.2014.09.005>. URL: <https://www.sciencedirect.com/science/article/pii/S0893608014002159> (cit. on p. 37).
- [57] Emad Barsoum, Cha Zhang, Cristian Canton Ferrer, and Zhengyou Zhang. «Training deep networks for facial expression recognition with crowd-sourced label distribution». In: *Proceedings of the 18th ACM International Conference on Multimodal Interaction*. 2016, pp. 279–283 (cit. on p. 37).
- [58] Michael J Lyons, Shigeru Akamatsu, Miyuki Kamachi, Jiro Gyoba, and Julien Budynek. «The Japanese female facial expression (JAFFE) database». In: *Proceedings of third international conference on automatic face and gesture recognition*. 1998, pp. 14–16 (cit. on p. 37).
- [59] Niki Aifanti, Christos Papachristou, and Anastasios Delopoulos. «The MUG facial expression database». In: *11th International Workshop on Image Analysis for Multimedia Interactive Services WIAMIS 10*. IEEE. 2010, pp. 1–4 (cit. on p. 37).
- [60] Oliver Langner, Ron Dotsch, Gijsbert Bijlstra, Daniel HJ Wigboldus, Skyler T Hawk, and AD Van Knippenberg. «Presentation and validation of the Radboud Faces Database». In: *Cognition and emotion* 24.8 (2010), pp. 1377–1388 (cit. on p. 37).
- [61] Abhinav Dhall, Roland Goecke, Simon Lucey, and Tom Gedeon. «Static facial expression analysis in tough conditions: Data, evaluation protocol and benchmark». In: *2011 IEEE international conference on computer vision workshops (ICCV workshops)*. IEEE. 2011, pp. 2106–2112 (cit. on p. 37).

- [62] Natalie C Ebner, Michaela Riediger, and Ulman Lindenberger. «FACES—A database of facial expressions in young, middle-aged, and older women and men: Development and validation». In: *Behavior research methods* 42.1 (2010), pp. 351–362 (cit. on p. 37).
- [63] Wikipedia. *Facial Action CodingSystem*. https://en.wikipedia.org/wiki/Facial_Action_Coding_System (cit. on pp. 38, 39).
- [64] Atul Balaji. *Real-time Facial Emotion Detection using deep learning*. <https://github.com/atulapra/Emotion-detection> (cit. on pp. 40, 41).
- [65] Stephen Welch Martin Cheung. *An application that detects facial emotion in real-time using Keras/Tensorflow and OpenCV in C++*. <https://github.com/martycheung/CppND-Facial-Emotion-Recognition> (cit. on p. 42).
- [66] Jacopo Sini. *MDPI-Sensors—IOT-Education-ReactionTimeTool*. <https://github.com/JacopoSini/MDPI-Sensors---IOT-Education-ReactionTimeTool> (cit. on pp. 46, 47).
- [67] stdev. *stdev*. <https://www.geeksforgeeks.org/python-statistics-stdev/> (cit. on p. 48).