DECEPTION DETECTION BY ANALYSING HUMAN PHYSIOGNOMY USING FACIAL RECOGNITION IN WINDOWS OS

Team:

Mohamed Afri (2021503029)

Shivani Suresh (2021503050)

Mugundh J B (2021503524)

Shyamala R B (2021503558)

INTRODUCTION

- Detecting deception is a challenge, and technology is being explored to automate the process and make the process more accurate.
- One approach is to analyze human physiognomy, i.e., facial features using facial recognition technology and detect micro-expressions that indicate deception.
- In this project, we will explore the potential and challenges of this approach, as well as its ethical implications.

PROBLEM STATEMENT

- Despite the importance of detecting deception in social professions such as teaching and psychology, current methods rely heavily on human interpretation and are prone to error.
- There is a need for an automated deception detection system that is more accurate and reliable, and that eliminates psychological factors to provide more precise results.
- The system should provide a straightforward indication of deception almost immediately, without significant processing time.

OBJECTIVES

- **Beneficial to many professionals:** To implement a favorable deception detection system to aid social professions like teachers, psychologists, etc.
- **Improved detection:** Automation of deception detection is useful in minimizing errors due to human intervention.
- **Revamped algorithm:** Uses the characteristic facial features like eyes, smile and facial movements to detect anomaly, hence eliminating psychological factors to give more precise results.
- **Straightforward indication:** Conclusion of detection will be indicated to the user almost immediately without much processing time.

LITERATURE SURVEY

S No.	Title of the paper	Name of the Journal and published year	Proposed Work	Proposed algorithm/ Methodology	Limitations
1	Lie Detection Using Image Processing	IEEE Transaction Advanced Computing and Communication Systems 2015	Lies are associated with a significant decrease in eye blinks. It is directly followed by an increase in eye blinks after the lie has been told.	The lie detection mechanism is based upon the experimentally proven hypothesis i.e., lies are associated with an increase in eye blinks when the cognitive demand ceases, after the lie has been told.	Cannot be used on fatigued drivers, Parkinson's disease etc. as they show the symptom of excessive blinking.
2	Head Movement analysis in Lie detection	IEEE Transaction Grid, Cloud and high performance computing 2015	Application was built in order to detect the head movement and head position by performing a frame-to-frame analysis on a color video stream.	When a suspect presents an affirmative situation, but shakes its head in a disapproval manner and the other way around.	Additional information Like voice, gaze & so on are required in addition to head movement.
3	Lie Detector With The Analysis Of The Change Of Diameter of Pupil & The Eye Movement	IEEE Transaction IOTAIS 2018	Analysis of eyes, namely with the object of eye tracking and eye pupil diameter changes by using method of Wavelet Transform. Result determination is done using a Decision Tree.	Eye movement towards the right means as movement that showed the lie because the right brain is used to imagine and think.	Accuracy of the lie detector is comparatively low
4	Lie Detector With Analysis Pupil Dilation And Eye Blinks Analysis Using Hough Transform And Decision Tree	IEEE Transaction ICCEREC 2015	Analysis of eyes, namely with the object of eye tracking and eye pupil diameter changes by using method of Hough Transform. Result determination is done using a Decision Tree.	The proposed algorithm uses the thesis of a person who is lying's pupil will turn towards the right.	Accuracy of the lie detector is comparatively low

LITERATURE SURVEY

S No.	Title of the paper	Name of the Journal and published year	Proposed Work	Proposed algorithm/ Methodology	Limitations
5	Lie Detector with Eye Movement and Eye Blinks Analysis Based on Image Processing using Viola-Jones Algorithm.	IEEE Transaction IOTAIS 2021	The proposed work uses The Viola-Jones method, which can detect faces, eye images, and changes in the eyes. Compared with fuzzy logic, this method could give more than 50% of accuracy.	The proposed algorithm involves detecting the face and eyes using the Viola-Jones algorithm, and analyzing the data to determine whether the individual is telling the truth.	Accuracy is Comparatively Low as change of pupil diameter and eye movement are theOnly factors taken into account.
6	Event Related Potential (ERP) based Lie Detection using a Wearable EEG headset	IEEE Transaction IBCAST 2019	A commercial off the shelf wearable EEG headset is used to record brain signals in an information concealment testing environment.	A machine learning based classification algorithm is developed using a wearable EEG headset and used with properly designed experiment to detect information concealment.	Limited availability of EEG Headset to record brain signals.
7	Vgg-16 And Vgg-19 Architecture Models In Lie Detection Using Image Processing.	IEEE Transaction ITISEE 2022	A collection of facial image data that is classified as honest or lying data is presented in this study.	Uses Deep Learning, VGG-16 and VGG-19, Augmentation, Convolutional Neural Networks	Highly sensitive to small changes which might be a disadvantage in lie detection.

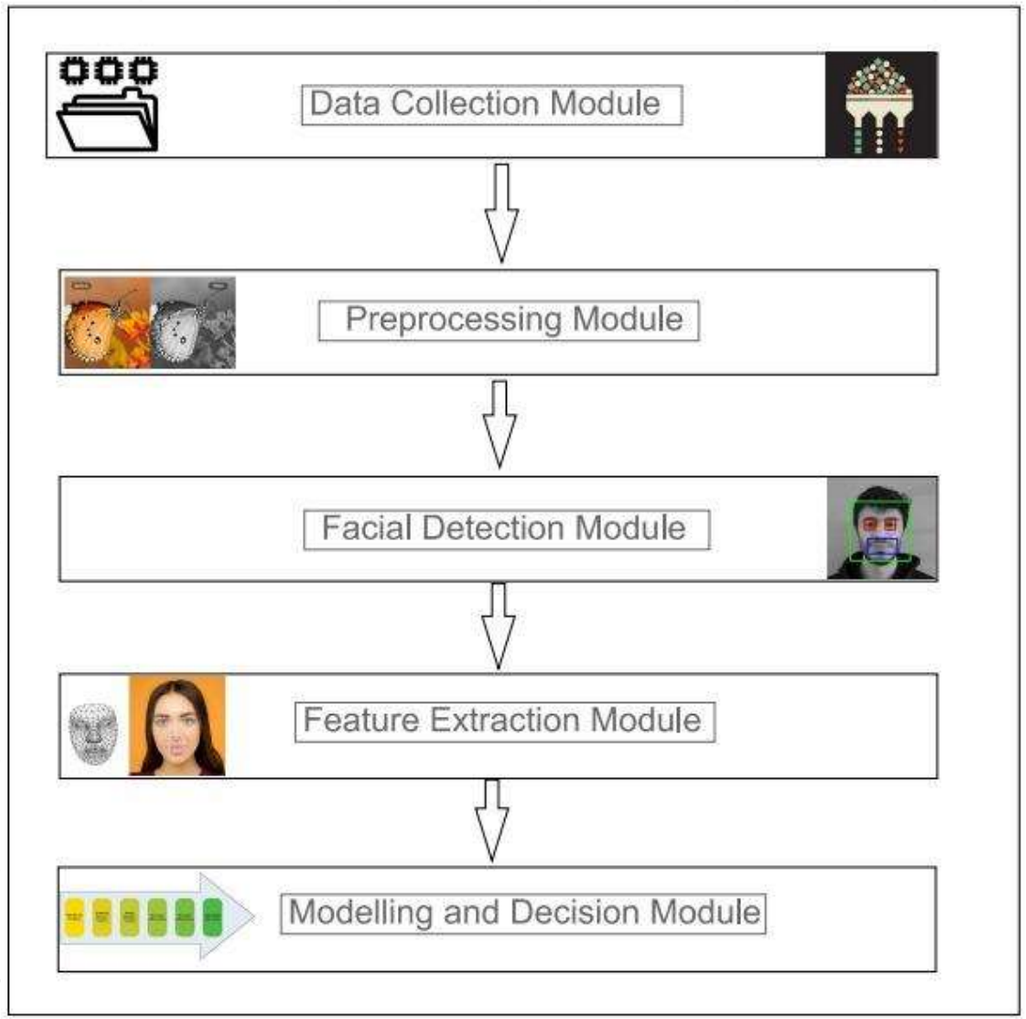
LITERATURE SURVEY

S No.	Title of the paper	Name of the Journal and published year	Proposed Work	Proposed algorithm/ Methodology	Limitations
8	A Novel Face Recognition-based Privacy Protection System for Mobile Devices	IEEE Transaction IOTAIS 2019	This paper proposes a face recognition-based privacy protection system that can be used to secure private files on mobile devices.	The system extracts facial features from the user's face and compares them to the features stored in the database.	The limitations of this system include the need for high-quality face images, and limited accuracy in low-light conditions.
9	FaceVault: A Facial Recognition-based Authentication System for Mobile Devices	IEEE Transaction Parallel and Distributed Systems 2019	This paper presents a facial recognition-based authentication system called FaceVault that can be used to protect private files on mobile devices.	The system uses deep learning algorithms to extract facial features from the user's face and compares them to the features stored in the database.	The limitations of this system include limited accuracy in low-light conditions.
10	Facial Recognition-based Access Control for Secure Personal Data Storage	IEEE Transaction Parallel and Distributed Systems 2019	This paper proposes a facial recognition-based access control system that can be used to secure personal data storage on mobile devices	The system captures the user's face and compares it to the stored facial features to grant access.	The limitations of this system include the need for high-quality face images.

PROPOSED WORK

- Ambiguities involved in using only certain facial features are removed by taking into account multiple facial characteristics like eyes, smile, and facial movements.
- The proposed system does not require additional information other than those analyzed by the detector.
- Psychological factors are not considered for evaluation in order to improve accuracy.

MECHANISM/MODULE



MECHANISM/MODULE

Data Collection Module:

This module involves collecting images or videos of human subjects that will be analyzed by the system.

The data can be obtained from various sources such as online databases, social media platforms, or by capturing images using a camera.

Preprocessing Module:

This module involves cleaning and transforming the raw image into a format that can be used for analysis.

This may include resizing, cropping, or normalizing the images.

MECHANISM/MODULE

Face, Eye and Smile Detection Module:

This module uses Haar cascades to detect and localize faces in the images or videos.

This involves applying a pre-trained classifier to search for facial features such as eyes, nose, and mouth.

Haar cascades is used to detect and localize the position of the eyes within the detected face regions; it is also used to detect and localize the position of the mouth and identify whether the person is smiling or not.

MECHANISM/MODULE

Feature Extraction Module:

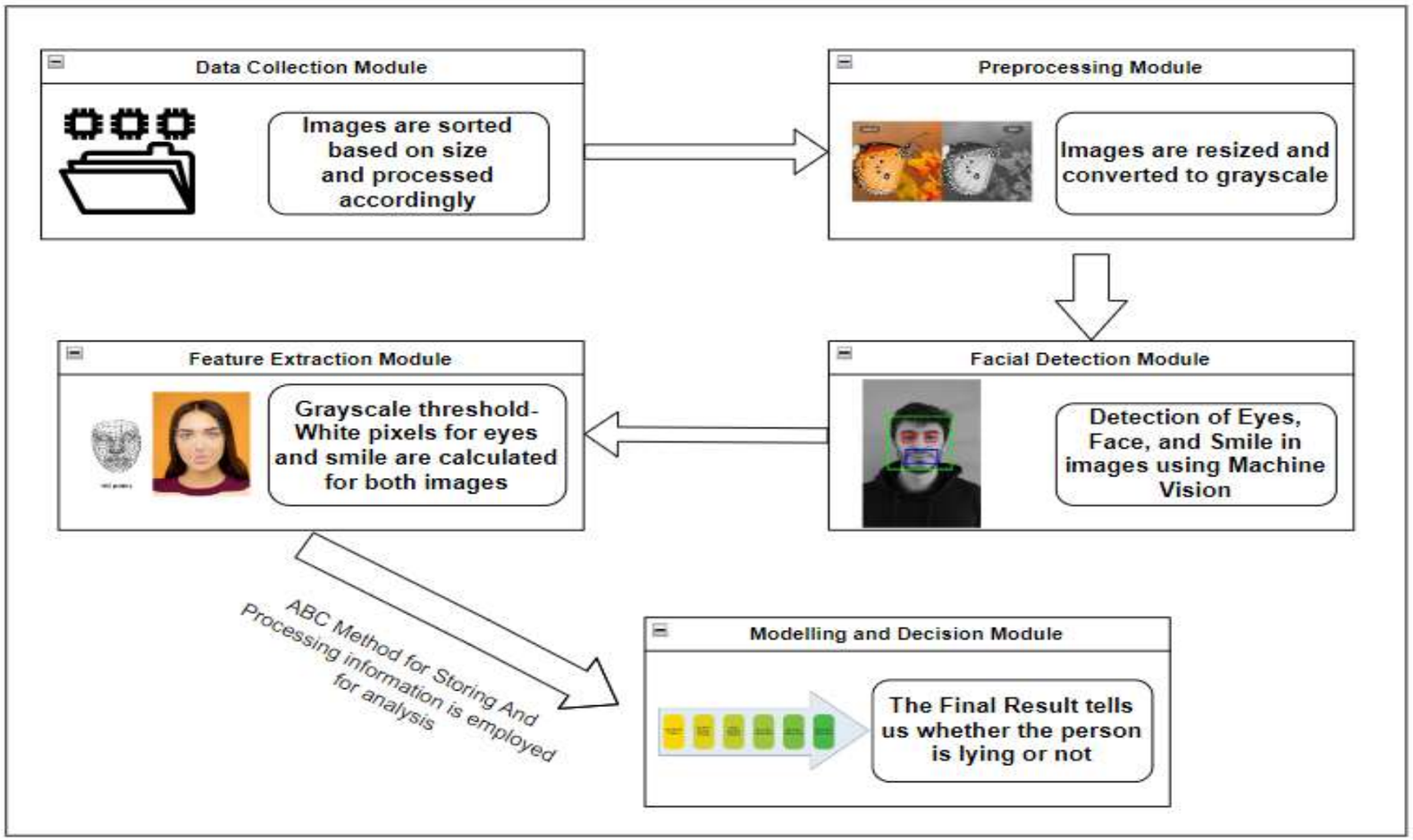
This module involves extracting relevant features from the detected face, eyes, and smile regions, such as the size, position, or movement of these features.

Modeling and Decision Module:

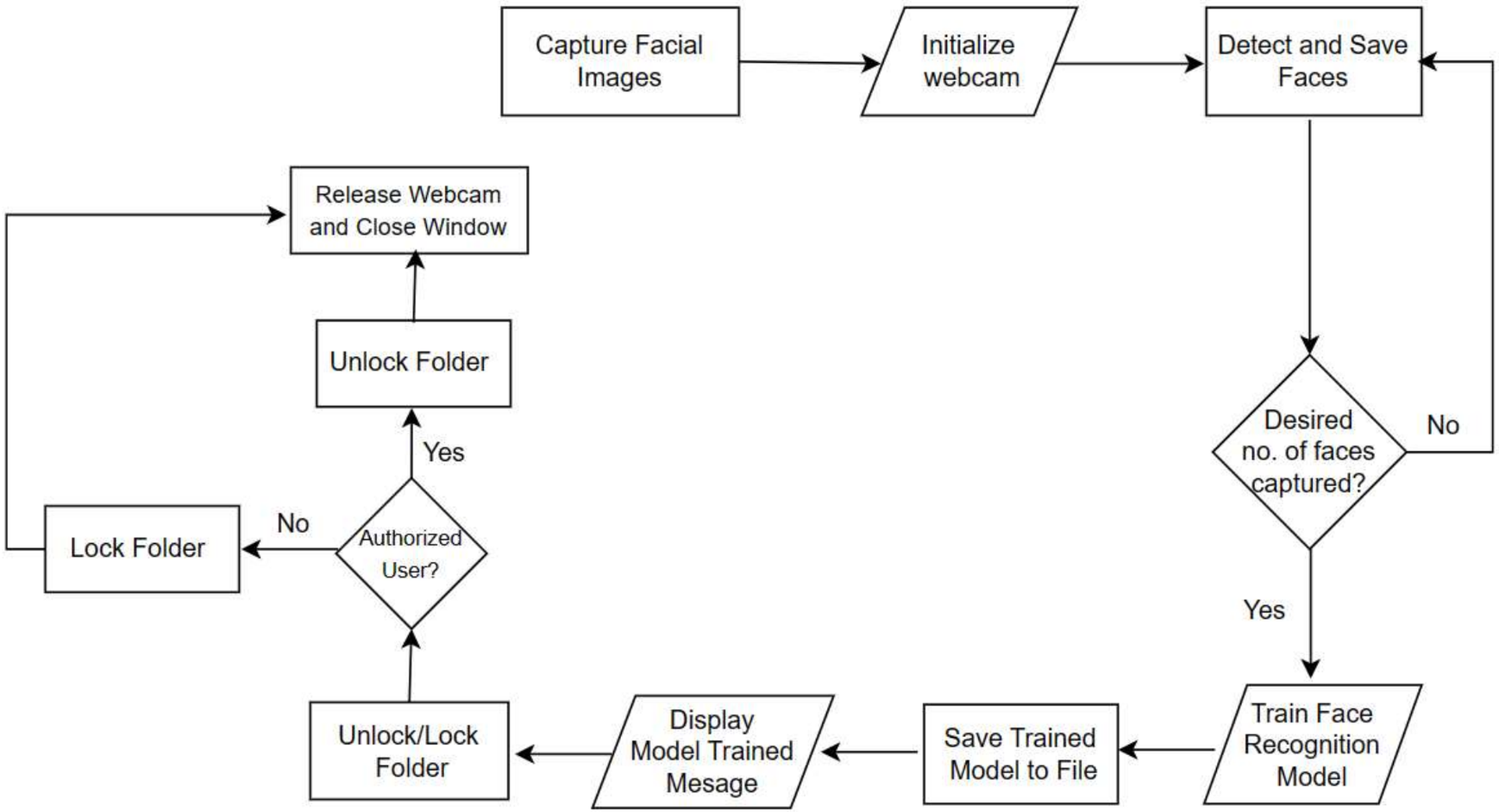
This module involves using statistical models or machine learning algorithms to analyze the extracted features and classify the person's behavior as truthful or deceptive.

The final decision is based on the output of the above step, such as determining whether a person is likely to be deceptive or not. This decision is made by the automated system.

ARCHITECTURE DIAGRAM FOR DeceptionDetectMV



ARCHITECTURE DIAGRAM FOR FaceUnlockTech



PROJECT COMPONENTS

01

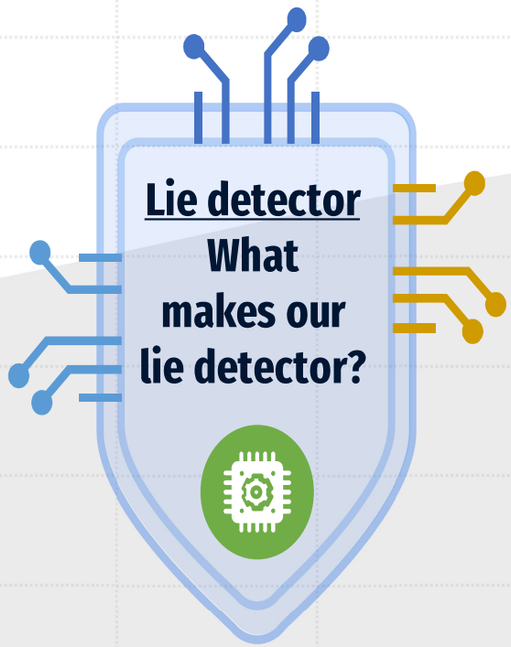
Controlled Environment

Lighting, angle, distance

02

Machine Vision

Techniques Learned



03

ABC Method

Finding the unknown using previous information

04

Haar Cascades

Face Detection, eye detection, mouth/smile detection

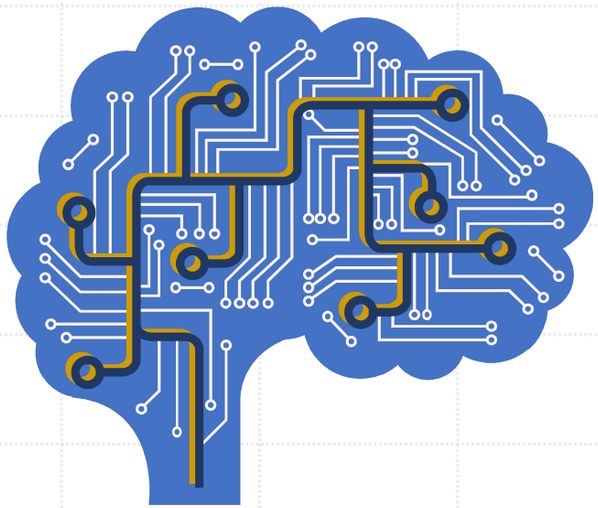
HAAR CASCADES – THE BACKBONE

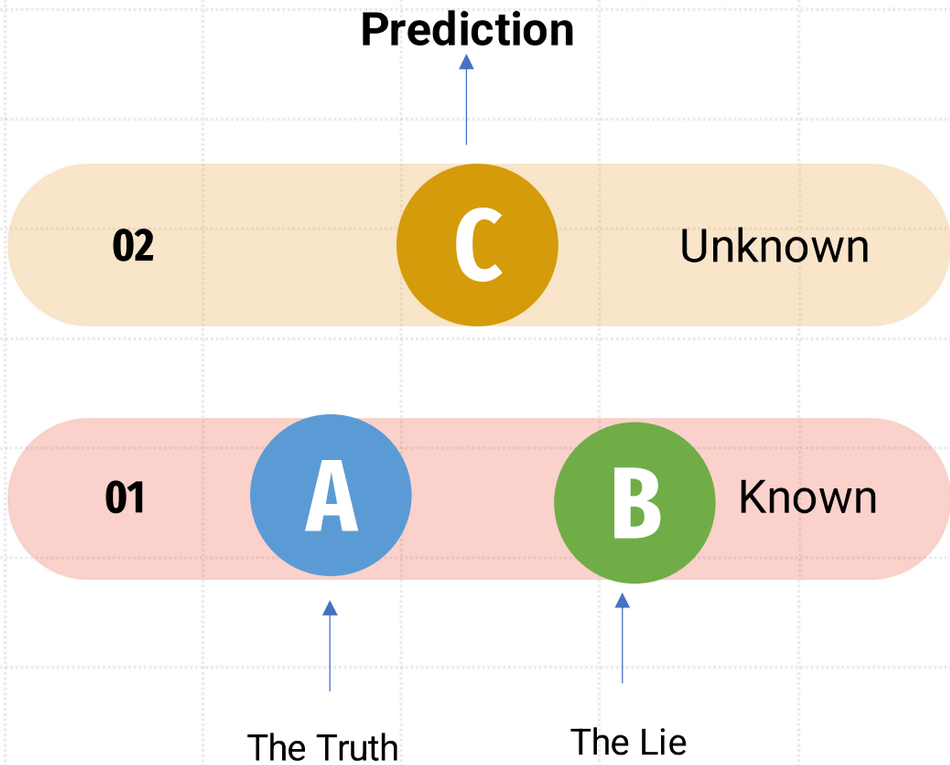
Face Detection	The encapsulating box
Eye Detection	The tricky Component
Smile Detection	The naïve Component

ABC METHOD

Depiction

Using A and B, find C





PICTORIAL REPRESENTATION OF FindTheDeceit

When the person is telling the truth: shrinking eyes + smaller smile

When the person is lying wide eyes + bigger smile

01

Step 1

Import the images from sorted folder – Loads detectors from local opencv library

02

Step 2

Detect the face – Detect the eyes in the upper half of the face – the smile in the bottom half

03

Step 3

Do a binary threshold – Calculate white pixels and smile size

04

Step 4

Present the Output

FindTheDeceit

- Start
- The images to be processed are stored in a folder.
- They are then sorted based on their size.
- The file having the smallest size is processed first, since it takes a shorter time for process completion (similar to the shortest job first algorithm).

Image segregation

- Image processing is done using a feature-based object detection (Haar cascades).
- The machine learning based approach is trained on lots of positive and negative images for detection.
- The face, eyes and size of the smile is determined this way.

Image processing

- The processed information is then analysed using a set of parameters to determine if a person is deceptive or not.
- End

FindTheDeceit

Input : *x,y.*
Output: *Truth or Lie*

Step 1: Start

Step 2: Load x and y

Step 3: $x1 = h(x);\ y1 = h(y)$

Step 4: $x2 = resize(x1); y2 = resize(y1)$

Step 5: $x3 = ccface(x2); y3 = ccface(y2)$

Step 6: $upper\_half\_faceArea\_x3 = img[fY3: fY3 + int(fH3/2), fX3: fX3 + fW3]$

$upper\_half\_faceArea\_y3 = img[fY3': fY3' + int(fH3'/2), fX3': fX3' + fW3']$

Step 7: $bottom\_half\_faceArea\_x3 = img[fY3 + int(fH3/2): fY3 + fH3, fX3: fX3 + fW3]$

$bottom\_half\_faceArea\_y3 = img[fY3' + int(fH3'/2): fY3' + fH3', fX3': fX3' + fW3']$

FindTheDeceit(Contd.)

Step 8: $x4 = cceyes(upper\_half\_faceArea\_x3); y4 = cceyes(upper\_half\_faceArea\_y3)$

Step 9: $x5 = ccsmile(bottom\_half\_faceArea\_x3); y5 = ccsmile(bottom\_half\_faceArea\_y3)$

Step 10: $x6 = bbeyes(x4); y6 = bbeyes(y4)$

Step 11: $x7 = bbsmile(x5); y7 = bbsmile(y5)$

Step 12: $x8 = bbface(x3); y8 = bbface(y3)$

Step 13: $pixels_{eyes1} = calculate\_pixels(x6); pixels_{eyes2} = calculate\_pixels(y6)$

Step 14: $pixels_{smile1} = calculate\_pixels(x7); pixels_{smile2} = calculate\_pixels(y7)$

Step 15: $If \;\; pixels_{smile1} < pixels_{smile2} \; and \; pixels_{eyes1} < pixels_{eyes2}, \;\; print \; lie.$

Step 16: $If \;\; pixels_{smile1} > pixels_{smile2} \; and \; pixels_{eyes1} > pixels_{eyes2}, \;\; print \; truth.$

FindTheDeceit-Algorithm Explanation

- Let variable ‘x’ represent the image with neutral expression and variable ‘y’ represent the image with change in expression.
- Let h(x) denote conversion of the image ‘x’ to grayscale and application histogram equalization to enhance contrast.
- Let resize(x) denote resizing the image ‘x’ to a smaller size.
- Let ccface(x), cceyes(x), and ccsmile(x) be cascade classifiers for face, eyes, and smiles detection in image ‘x’ created using the Haar cascades.
- Let x2 and y2 be resized images of x1 and y1 obtained using resize(x).
- Let x3 and y3 be images obtained after applying ccface(x) on x2 and y2.
- Let x4 and y4 be images obtained after applying cceyes(x) on upper_half_faceAreax3 and upper_half_faceAreay3.

FindTheDeceit-Algorithm Explanation(Contd.)

- Let x5 and y5 be images obtained after applying ccsmile(x) on bottom_half_faceAreax3 and bottom_half_faceAreay3.
- Let bbeyes(x), bbface(x) and bbsmile(x) denote drawing bounding box around eyes, face and smile respectively.
- Let x6 and y6 be images obtained after applying bbeyes(x) on x4 and y4.
- Let x7 and y7 be images obtained after applying bbsmile(x) on x5 and y5.
- Let x8 and y8 be images obtained after applying bbface(x) on x3 and y3.
- Let calculate_pixels(x) denote calculating white pixels in enclosed box in an image 'x'.
- Let $pixels_{eyes1}$ and $pixels_{eyes2}$ be the values obtained after applying calculate_pixels(x) on x6 and y6.
- Let $pixels_{smile1}$ and $pixels_{smile2}$ be the values obtained after applying calculate_pixels(x) on x7 and y7.

FaceUnlockTech-Algorithm

Step 1: Get the name of the folder to lock from the user.

Step 2: Load the pre-trained face recognition model.

Step 3: Initialize the webcam.

Step 4: Start a loop that runs indefinitely.

Step 5: Get a frame from the webcam.

Step 6: Convert the frame to grayscale.

Step 7: Detect faces in the frame.

Step 8: If an authorized user is detected, unlock the folder.

Step 9: If an authorized user is not detected, lock the folder.

Step 10: Check for key press events.

Step 11: Release the webcam and close the window.

HARDWARE REQUIREMENTS

Hardware:

A computer with the following specifications:

Processor :11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz 2.80 GHz

Installed RAM :16.0 GB (15.8 GB usable)

System type :64-bit operating system, x64-based processor

Webcam is optional.

OS:

Edition :Windows 11 Home Single Language

Version :22H2

OS build :22621.1413

SOFTWARE REQUIREMENTS

Software:

- Visual Studio Code with appropriate Python extension/ IDLE (Python 3.11 64-bit).
- OpenCV library.
- OS Module in Python.
- Numpy module in Python.
- The following .xml files for Haar Cascades:
  - haarcascade_eye.xml
  - haarcascade_frontalface_default.xml
  - haarcascade_smile.xml

IMPLEMENTATION DETAILS-FindTheDeceit

- Start by importing the necessary modules, including the OS module and OpenCV.
- Define a function to process images in a folder, which takes the folder path as an input parameter.
- Use the OS module to get a list of image files in the folder.
- Sort the list of image files based on size, so that smaller images are processed first.
- Loop through each image file in the list.
- Load the image using OpenCV's imread() function.
- Use Haar Cascades to perform face detection on the image. This involves loading the face detection classifier (haarcascade_frontalface_default.xml) and converting the image to grayscale. Then, use the detectMultiScale() function to detect faces in the image, and store the coordinates of the bounding boxes around each face.

IMPLEMENTATION DETAILS-FindTheDeceit (Cont'd)

- For each face detected in the image, use Haar Cascades to detect eyes and smiles. This involves loading the eye and smile detection classifiers (haarcascade_eye.xml and haarcascade_smile.xml), and using the detectMultiScale() function to detect eyes and smiles within the bounding box of each face.
- Calculate the number of white pixels in the smile region using OpenCV's threshold() function and calculate the size of the smile region using OpenCV's contourArea() function.
- Compare the detected features between the image with neutral expression and the image to be examined using the ABC method, where two known factors are employed to determine an unknown result. If the eyes have shrunk and the smile has gotten smaller, then conclude that the person is being truthful. If the eyes are enlarged and the smile has gotten bigger, then conclude that the person is being deceptive.
- Output the result of the analysis to the user in a dialog box.
- End the function

IMPLEMENTATION DETAILS-FaceUnlockTech

- Firstly, install the necessary dependencies (OpenCV and Tkinter) and import the modules (cv2, os ,numpy, shutl, tkinter)
- To set the authorized user's name and dataset folder path, assign the name to the variable `person_name` and concatenate it with the desired folder path to the variable `dataset_path`.
- Create the dataset folder if it doesn't exist using the command os.makedirs(dataset_path)
- Initialize the webcam and face counter using cv2.VideoCapture(0) and face_counter = 0
- Capture facial images by reading frames from the webcam.
- Release the webcam using cap.release() and close the window with the command cv2.destroyAllWindows().

IMPLEMENTATION DETAILS-FaceUnlockTech(cont'd)

- To load the training data, a function named load_training_data() is defined which iterates over the subfolders in the dataset folder and assigns a label ID to each subfolder. Finally, the function returns the collected faces, labels, and label IDs for further processing.
- Load the training data.
- To create and train the face recognition model, the following steps are performed:
  - First, the face recognizer object is created using `cv2.face.LBPHFaceRecognizer_create()`.
  - Then, the model is trained by passing the faces and corresponding labels as arguments to the `train()` function of the face recognizer.
  - After training, the trained model is saved to a file specified by `model_path` using the `save()` function of the face recognizer.
  - Lastly, a messagebox is displayed with the message "Face model trained" using `messagebox.showinfo()`.
- Create the main window for user input using Tkinter

IMPLEMENTATION DETAILS-FaceUnlockTech(cont'd)

- Create a label, entry field, and button for user input
- To handle a button click, define the function `handle_button_click()`. Inside this function, retrieve the user input from the entry field. If the user input is not empty, display a messagebox showing the user input. If the user input is empty, display an error messagebox.
- Run the main window loop and load the pre-trained face recognition model by calling the function cv2.face.LBPHFaceRecognizer_create().
- To lock or unlock a folder based on face recognition, the steps are as follows:
  - Initialize the webcam, detect faces in the captured frames, recognize the faces using the loaded model, and compare with the authorized user.
  - If there is a match, unlock the folder; otherwise, lock it.
  - Finally, release the webcam and close the window.

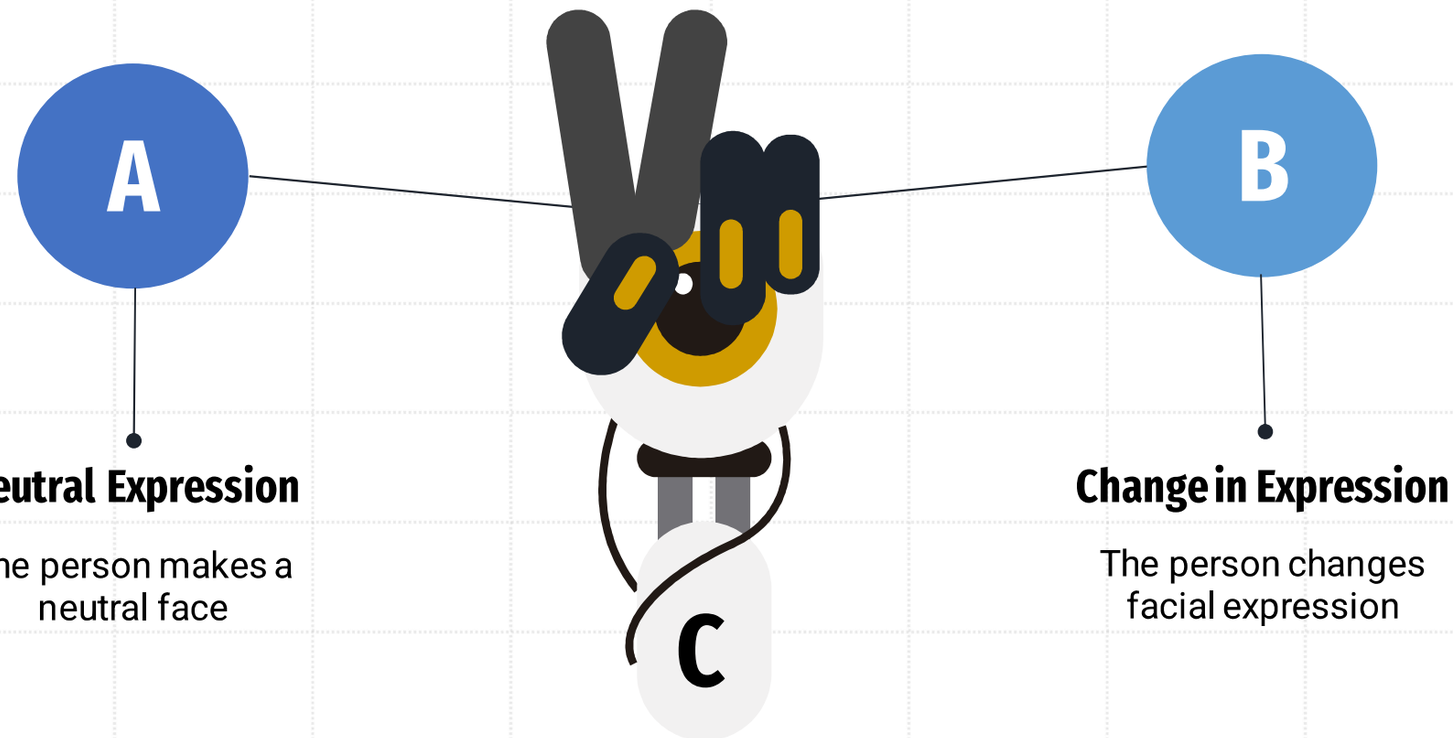
KEY FEATURES

- **Minimal Hardware Requirement:** Requires only a system to process images for deception detection.
- **Efficient Image Processing:** Images with minimal processing time are processed first to reduce the waiting time for subsequent images.
- **Accuracy:** Accuracy is enhanced as only the least ambiguous facial features are examined for deception detection.
- **Quick Result:** The processing time for each image is low and results are produced almost instantly.
- **Extensibility:** The proposed system allows room for multiple image processing.

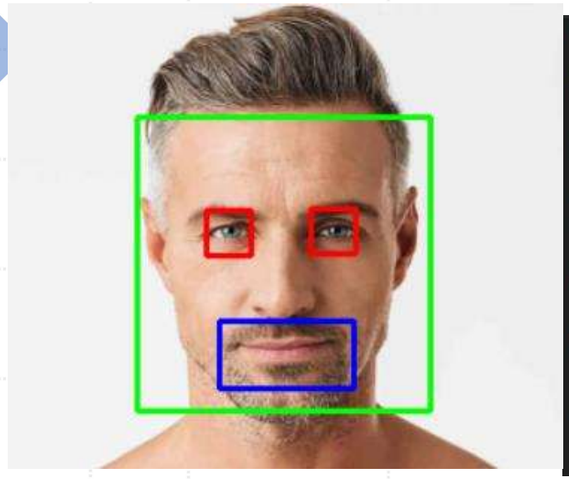
KEY FEATURES

- **Reliability:** The system has minimal down time hence increasing its reliability.
- **Precision:** Captures even small subtle facial movements like eye twitch,etc.
- **Economic:** The deception detection system requires minimal software and hardware requirement and is hence very economical.
- **User-friendly:** The user simply has to load the images to be processed on to a folder. The system requires no further input.
- **Dependability:** The results are arrived based on mathematical calculation and hence there is less room for false answers.

A SAMPLE TEST



IMPLEMENTATION SO FAR



```python
import cv2

# Load the cascade
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_def

# Read the input image
img = cv2.imread('input_image.jpg')

# Convert into grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Detect faces
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)

# Draw rectangle around the faces
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (0, 0, 255), 2)

# Display the output
cv2.imshow('img', img)
cv2.waitKey()
```

Face, Eyes and Smile detection

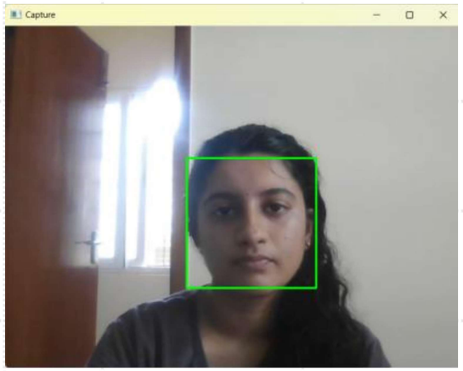
File sorting and processing (Implemented)

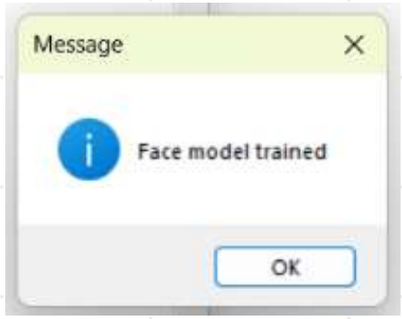
```python
import os
from os import listdir
import numpy as np
import cv2
# get the path/directory
face_detector=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
dir_name = "C:\\Shivani\\MIT\\Notes\\Semester_4\\OS_Project\\Trials"
# Get list of all files only in the given directory
list_of_files = filter(lambda x: os.path.isfile(os.path.join(dir_name, x)) and x.endswith('.jpg'),
        os.listdir(dir_name))
# Sort list of file names by size
list_of_files = sorted( list_of_files,
                        key =  lambda x: os.stat(os.path.join(dir_name, x)).st_size)
# Iterate over sorted list of file names and
# print them one by one along with size
for file_name in list_of_files:

        file_size  = os.stat(file_name).st_size
        print(file_size, ' -->', file_name)
        img = cv2.imread(file_name)
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces = face_detector.detectMultiScale(gray, 1.3, 5)
        for (x,y,w,h) in faces:
            cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)

        cv2.imshow('img',img)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
        #print(images)
```

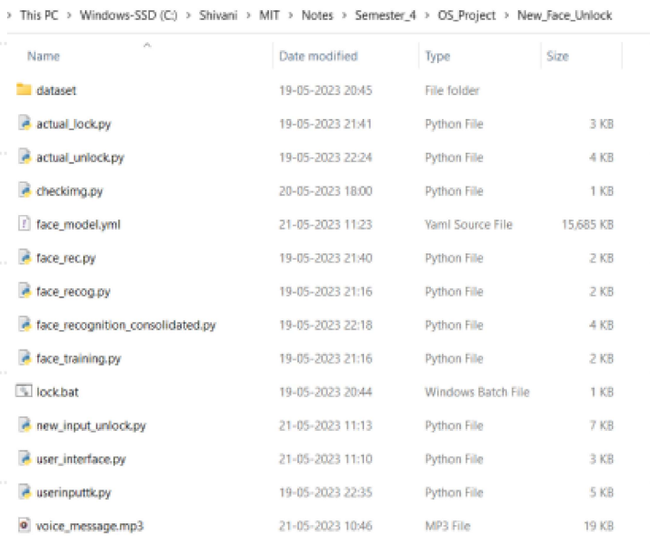

```python
import cv2

# Load the cascades
face_cascade = cv2.CascadeClassifier('haarcascade_frontalfa
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

# Read the input image
img = cv2.imread('person2.jpg')

# Convert into grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Detect faces
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)

# Draw rectangle around the faces and eyes
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (0, 0, 255), 2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img[y:y+h, x:x+w]
    eyes = eye_cascade.detectMultiScale(roi_gray)
    for (ex, ey, ew, eh) in eyes:
        cv2.rectangle(roi_color, (ex, ey), (ex+ew, ey+eh), (255, 0, 0), 2)

# Display the output
cv2.imshow('img', img)
cv2.waitKey()
```

Implementation results

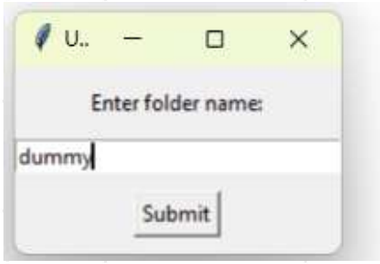
IMPLEMENTATION SO FAR

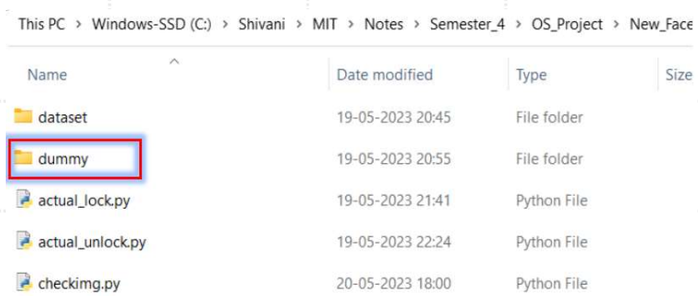


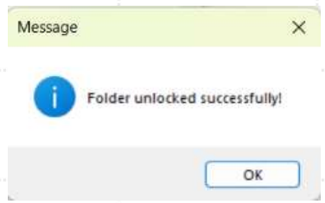




Initially folder dummy is hidden(locked).







Folder was unlocked.

File lock using face recognition (Implemented)

REFERENCES

1) B. Singh, P. Rajiv and M. Chandra, "Lie detection using image processing," 2015 IEEE Transactions on Advanced Computing and Communication Systems, pp. 1-5, doi: 10.1109/ICACCS.2015.7324092.in

2) D. -I. Noje and R. Malutan, "Head movement analysis in lie detection," 2015 IEEE Transaction on Grid, Cloud & High Performance Computing in Science , pp. 1-4, doi: 10.1109/ROLCG.2015.7367432.

3) A. Thompson, "The Cascading Haar Wavelet Algorithm for Computing the Walsh–Hadamard Transform," in IEEE Signal Processing Letters, vol. 24, no. 7, pp. 1020-1023, July 2017, doi: 10.1109/LSP.2017.2705247.

REFERENCES

4) S. Sumriddetchkajorn, A. Somboonkaew, T. Sodsong, I. Promduang, N. Sumriddetchkajorn and T. Prada-in, "Simultaneous Analysis of Far Infrared Signals From Periorbital and Nostril Areas for Nonintrusive Lie Detection: Performance From Real Case Study," in Journal of Lightwave Technology, vol. 33, no. 16, pp. 3406-3412, 15 Aug.15, 2015, doi: 10.1109/JLT.2014.2371466.

5) A. A. Perdana et al., "Lie Detector with Eye Movement and Eye Blinks Analysis Based on Image Processing using Viola-Jones Algorithm," 2021 IEEE Transaction on Internet of Things and Intelligence Systems (IoTaIS) pp. 203-209, doi: 10.1109/IoTaIS53735.2021.9628413.

6) S. Anwar, T. Batool and M. Majid, "Event Related Potential (ERP) based Lie Detection using a Wearable EEG headset," 2019 IEEE Transaction on Applied Sciences and Technology (IBCAST) pp. 543-547, doi: 10.1109/IBCAST.2019.8667131.

REFERENCES

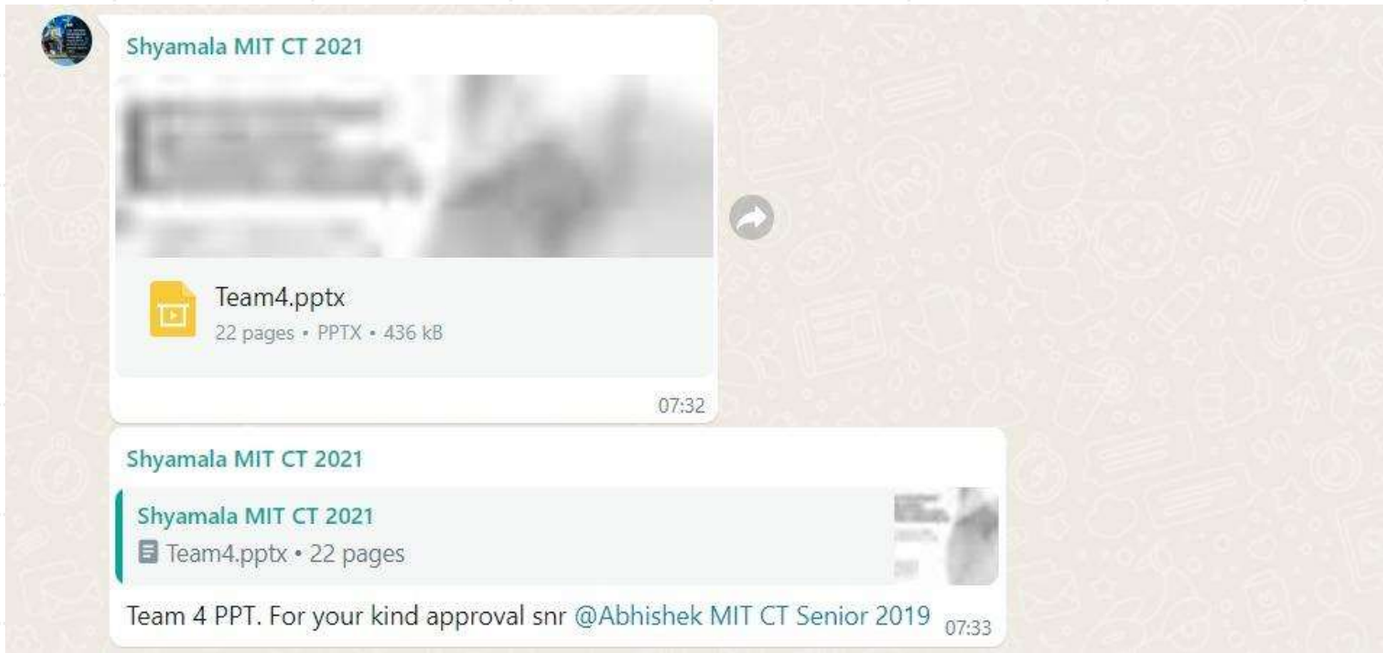
7) D. Kusumawati, A. A. Ilham, A. Achmad and I. Nurtanio, "Vgg-16 And Vgg-19 Architecture Models In Lie Detection Using Image Processing," 2022 IEEE Transaction on Information Technology, Information Systems and Electrical Engineering, pp. 340-345, doi: 10.1109/ICITISEE57756.2022.10057748.

8) T. A. Wibowo, M. Nasrun and C. Setianingsih, "Lie Detector With Analysis Pupil Dilation And Eye Blinks Analysis Using Hough Transform And Decision Tree," 2018 IEEE Transaction on Control, Electronics, Renewable Energy and Communication, 2018, pp. 172-178, doi: 10.1109/ICCEREC.2018.8712103.

9) Z. Labibah, M. Nasrun and C. Setianingsih, "Lie Detector With The Analysis Of The Change Of Diameter Pupil and The Eye Movement Use Method Gabor Wavelet Transform and Decision Tree," 2018 IEEE Transaction on Internet of Things and Intelligence System (IOTAIS) 2018, pp. 214-220, doi: 10.1109/IOTAIS.2018.8600918.

10) S. V. Fernandes and M. S. Ullah, "Use of Machine Learning for Deception Detection From Spectral and Cepstral Features of Speech Signals," in IEEE Access, vol. 9, pp. 78925-78935, 2021, doi: 10.1109/ACCESS.2021.3084200.

APPROVAL FROM MENTOR



THANK YOU!