FACE RECOGNITION AND LIVENESS DETECTION BASED ATTENDANCE SYSTEM

Team Members
Aditya Ashish Bakshi 20BCE1516
Anant Tater 20BAI1127
Aman Sharda 20BAI1180
Raghav Sharma 20BAI1084

Guide
Geetha S.

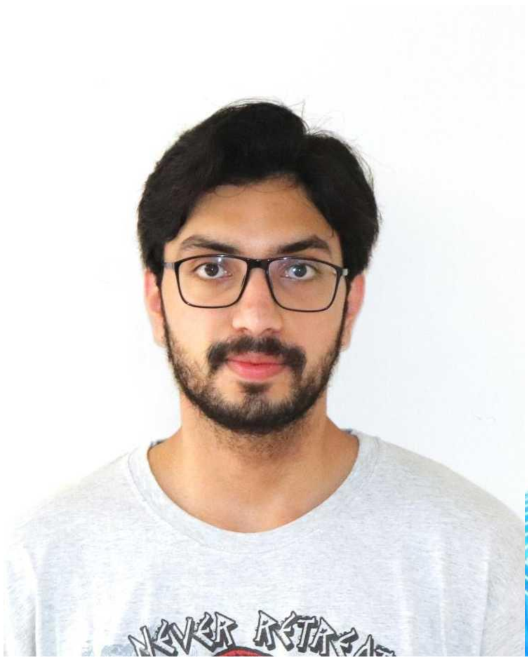
School name: School of Computing Science and Engineering

Faculty name (ERP No): S.Geetha, (50587)

Academic Year: 2023-2024(Fall Semester 23-24)

Title of the project: Face Recognition and Liveness Detection based Attendance system

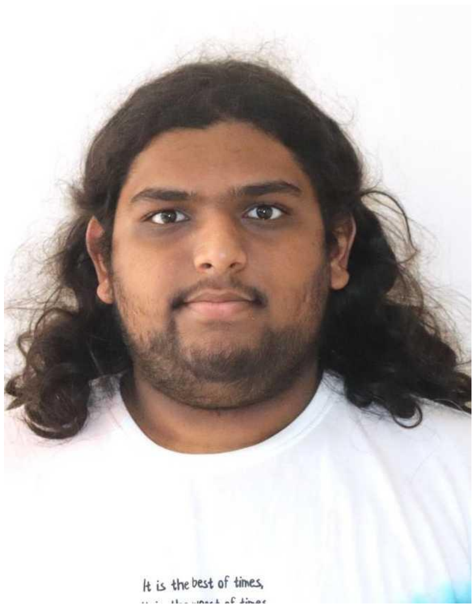
Number of students involved in this project: 4



20BAI1084

20BAI1127

20BAI1180



20BCE1516

Table of Content

Chapter 1: Introduction

1.1 Abstract

In the era of digital transformation, traditional methods of attendance tracking are becoming obsolete. This project aims to develop an innovative attendance system using face recognition and liveness detection technologies. The proposed system will capture and analyze facial features in real-time to identify individuals and mark their attendance.

The face recognition component will leverage machine learning algorithms to accurately identify individuals from a database of known faces. It will be robust to variations in lighting, pose, and facial expressions, ensuring high recognition accuracy.

The liveness detection component will add an additional layer of security to prevent spoofing attacks. It will distinguish between real faces and fake ones (e.g., photos, videos, or masks), ensuring that the attendance system cannot be easily manipulated.

The integration of these two technologies will result in a secure, efficient, and user-friendly attendance system. This system could have wide applications in various sectors, including education, corporate, and event management, revolutionizing the way attendance is recorded and managed.

1.2 Problem Definition

Attendance tracking is a critical operation in various sectors such as educational institutions, corporate offices, and event management. Traditional methods of attendance tracking, such as manual roll calls or sign-in sheets, are time-consuming, prone to errors, and can be easily manipulated.

With the advancements in technology, biometric systems have been introduced for attendance tracking. However, these systems often require physical contact, leading to hygiene concerns, especially in the post-pandemic world. Furthermore, these systems can be expensive to implement and maintain.

Face recognition technology offers a contactless alternative for attendance tracking. However, it is susceptible to spoofing attacks where someone tries to impersonate another individual by using their photo or video. This vulnerability can lead to false attendance records.

Therefore, the problem this project aims to address is to develop a secure, efficient, and user-friendly attendance system using face recognition and liveness detection technologies. The system should be able to accurately identify individuals from a database of known faces and distinguish between real faces and fake ones. The system should be robust to variations in lighting, pose, and facial expressions, and should prevent spoofing attacks effectively.

1.3 Motivation

- **Efficiency**: Traditional methods of attendance tracking are time-consuming and prone to errors. An automated system using face recognition can streamline this process, saving time and reducing errors.
- **Security**: Face recognition provides a level of security that is not possible with traditional methods. It can prevent unauthorized access and ensure that only registered individuals are marked present.
- **Hygiene**: In the post-pandemic world, contactless technologies are preferred for hygiene reasons. Face recognition is a contactless method, making it a safer alternative to biometric systems that require physical contact.
- **User-friendly**: Face recognition systems are generally user-friendly. Users do not need to carry any special equipment or cards - they just need to be present in front of the camera.

- **Spoofing Prevention**: The addition of liveness detection can prevent spoofing attacks, adding an extra layer of security to the system.

Chapter 2 Methodology

2.1 OpenCv2

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV2, a part of OpenCV, is used extensively in real-time image processing. It allows us to actively capture images from the camera and process the data to add attendance as people keep scanning.

2.2 Face_recognition

1. **Face Detection**: The first step in face recognition is detecting faces in an image. The face_recognition module provides a function face_locations that returns an array of bounding boxes of human faces in a image.

```
import face_recognition
image = face_recognition.load_image_file("your_file.jpg")
face_locations = face_recognition.face_locations(image)
```

2. **Face Encoding**: Once the faces are detected, the next step is to encode the facial features into a vector. The face_encodings function returns a list of 128-dimensional face encodings (one for each face in the image).

```
face_encodings = face_recognition.face_encodings(image)
```

3. **Face Recognition**: The compare_faces function compares a list of face encodings against a candidate encoding to see if they match.

```
known_image = face_recognition.load_image_file("known_person.jpg")
unknown_image = face_recognition.load_image_file("unknown.jpg")

known_encoding = face_recognition.face_encodings(known_image)[0]
unknown_encoding = face_recognition.face_encodings(unknown_image)[0]

results = face_recognition.compare_faces([known_encoding], unknown_encoding)
```

2.3 Liveness Detection

Convolutional Neural Networks (CNNs) are a type of deep learning model that are particularly effective at processing grid-like data, such as images. They have been extensively used for face liveness detection. Here's how they can be utilized:

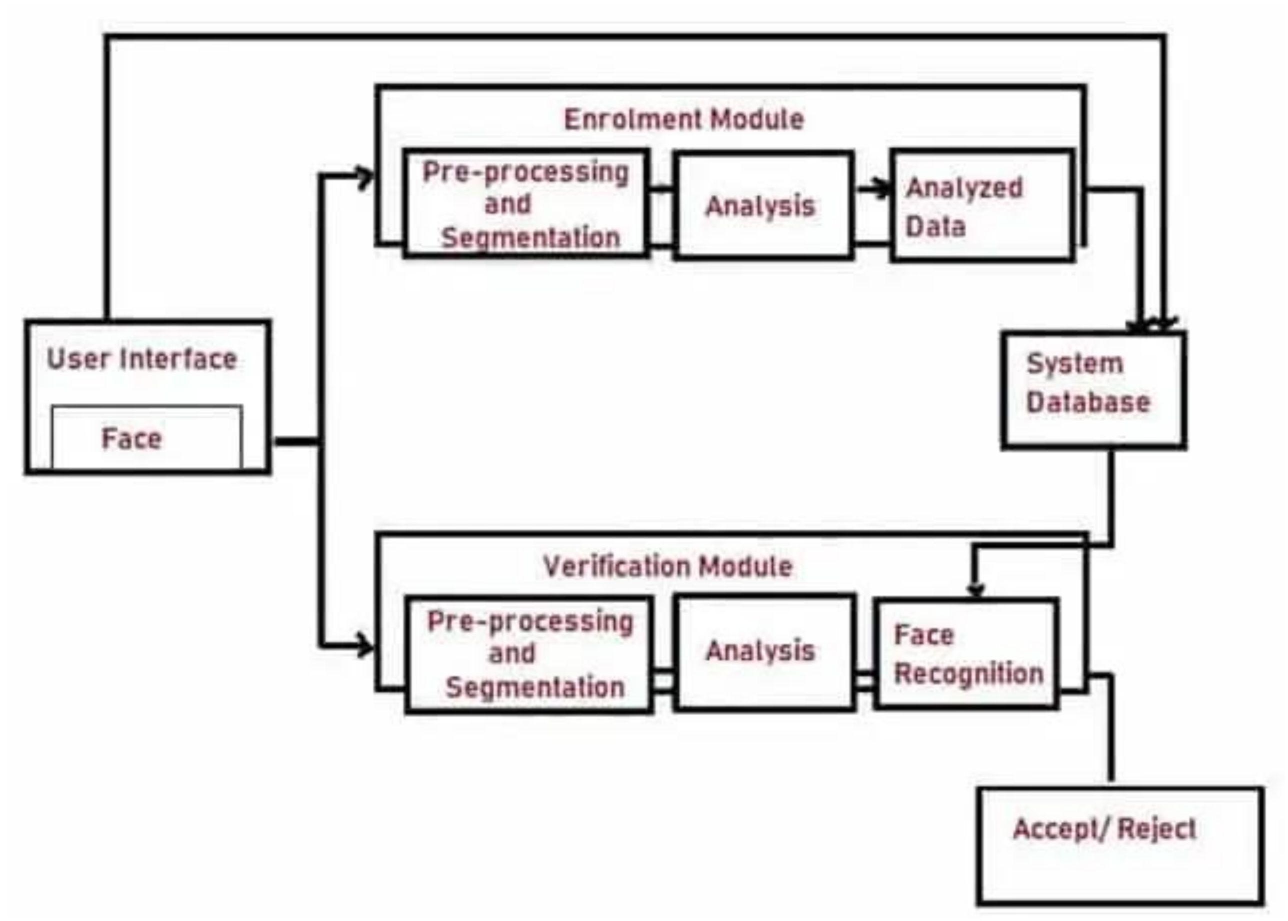
- **Feature Learning**: CNNs automatically learn hierarchical representations of input data. In the context of face liveness detection, a CNN can learn to recognize important features that distinguish real faces from fake ones. For instance, it could learn that certain patterns of pixels are more common in photos of faces than in real faces.
- **Training**: The CNN is trained on a dataset containing both real and spoof facial images. During training, the model learns to adjust its weights and biases to minimize the difference between its predictions and the actual labels. This is usually done using a variant of stochastic gradient descent.
- **Classification**: Once trained, the CNN can classify new images as real or fake. It does this by passing the image through the network, applying the learned transformations, and outputting a probability that the image is real.
- **Real-Time Processing**: CNNs can process images in real-time, making them suitable for liveness detection in face recognition systems. They can take as input the frames captured from a webcam and process them to detect and recognize faces.

Chapter 3 Architecture

3.1 Face Detection

The face detection has 3 main parts:

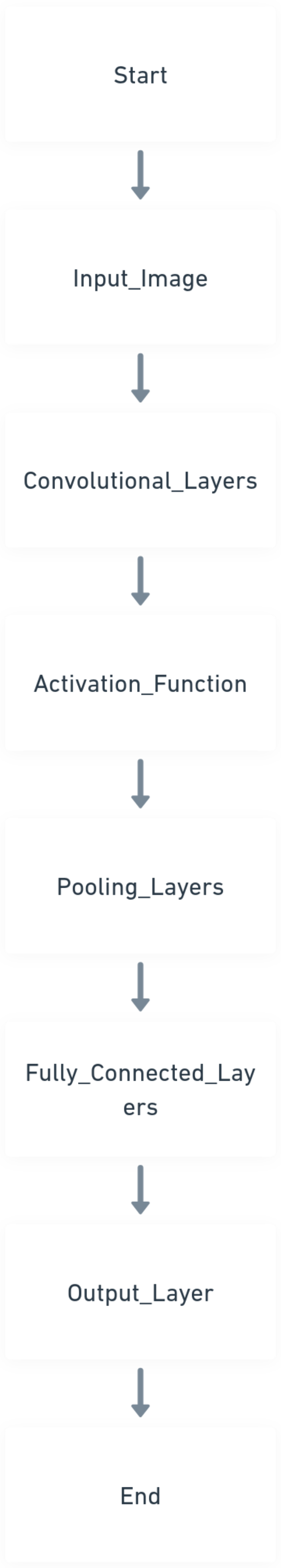
1. **Face Detection**: The system accepts the image as an input and checks if a 'Face' appears in the image and calculates its position on the image. The output of this stage is 'Patches' which contains 'Face' and face alignment is done which acts as a pre-processing stage for feature extraction.
2. **Feature Extraction**: Face patch is transformed into a set of fiducial points corresponding to their locations or it is transformed into vectors with specific dimension. This step involves extracting the unique features from the detected faces.
3. **Face Recognition**: This step includes recognition of face from the database. When the system receives a face image, it undergoes face detection and feature extraction process. Then, the features are compared with each face in the database using the nodal points on the face.



3.1 Anti Spoof

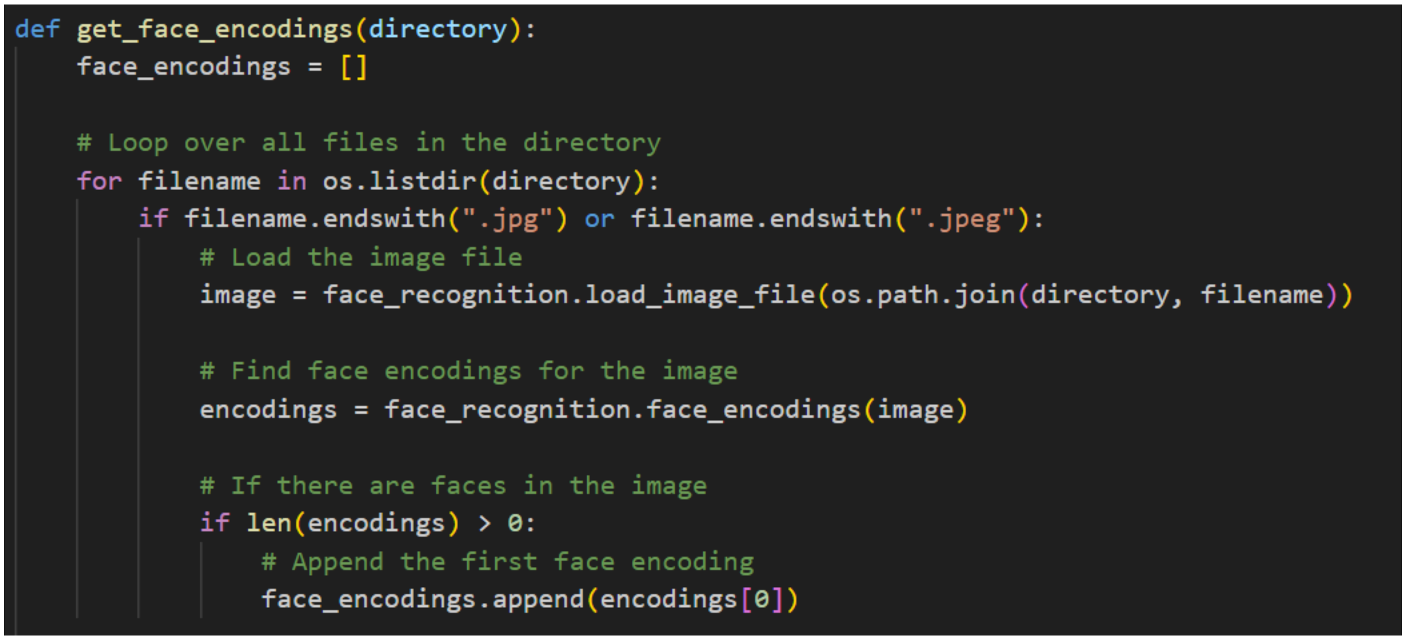
Sure, here’s a high-level overview of the technical architecture of Convolutional Neural Networks (CNNs) used for liveness detection:

1. **Input Layer**: This is where the network receives input in the form of an image or a frame from a video sequence.
2. **Convolutional Layers**: These layers apply a series of filters to the input data to create a feature map that represents the input image. The filters can detect low-level features such as edges and curves, as well as high-level features like shapes or objects.
3. **Activation Function**: After each convolutional layer, an activation function is applied. The most common activation function is the Rectified Linear Unit (ReLU), which introduces non-linearity into the model.
4. **Pooling Layers**: These layers reduce the spatial size of the feature map to decrease the computational complexity of the network, while preserving the most important features.
5. **Fully Connected Layers**: These layers take the output of the previous layers and flatten it into a single vector. This vector is then passed through one or more fully connected layers to map the input to the desired number of classes.
6. **Output Layer**: The final fully connected layer uses a softmax activation function to output a probability distribution over the classes. For liveness detection, there would be two classes: real and fake.
7. **Training**: The network is trained on a dataset containing both real and spoof facial images. During training, the model learns to adjust its weights and biases to minimize the difference between its predictions and the actual labels. This is usually done using a variant of stochastic gradient descent.
8. **Classification**: Once trained, the CNN can classify new images as real or fake. It does this by passing the image through the network, applying the learned transformations, and outputting a probability that the image is real


Start
Input_Image
Convolutional_Layers
Activation_Function
Pooling_Layers
Fully_Connected_Lay
ers
Output_Layer
End

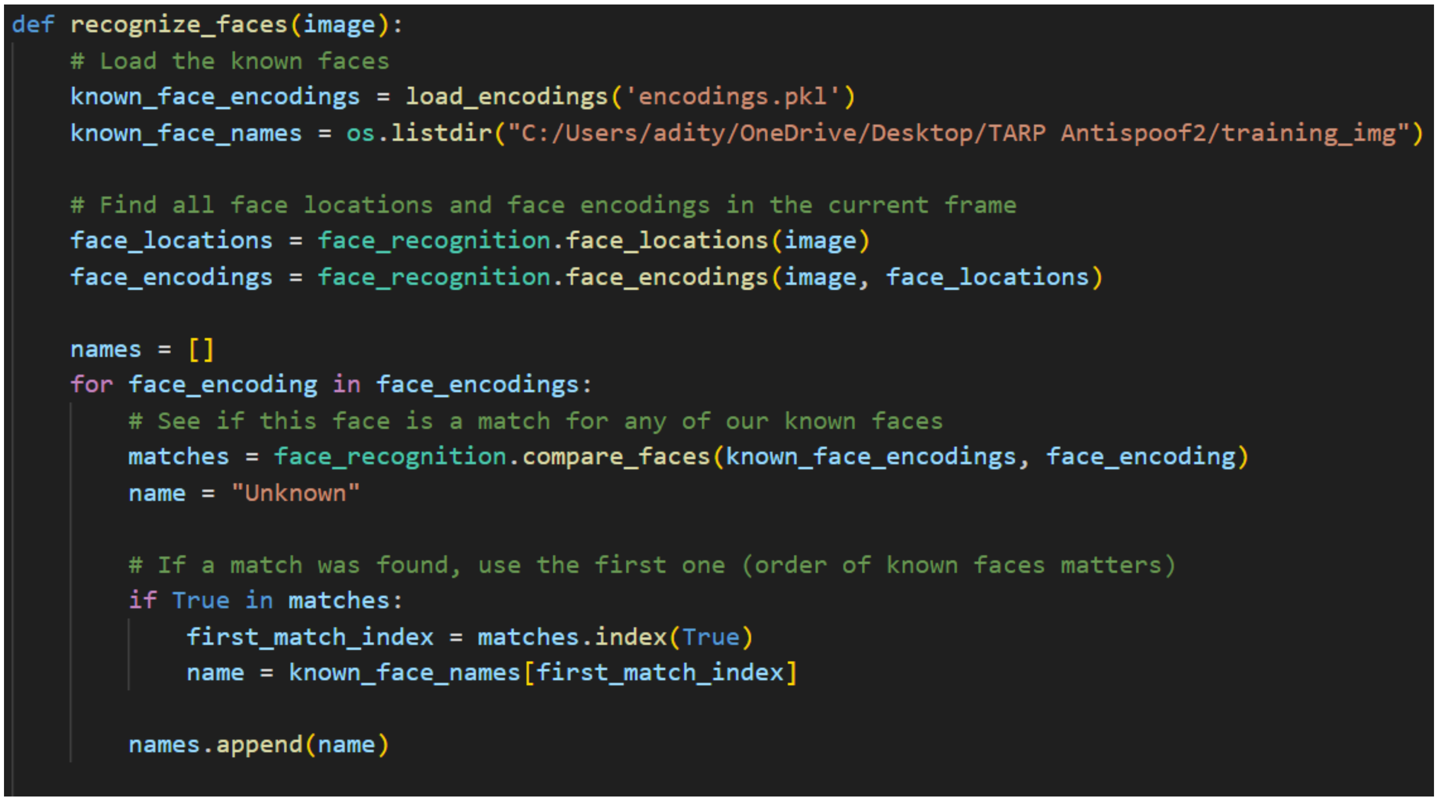
Chapter 4 Implementation

4.1 Creating face encodings based on training data:



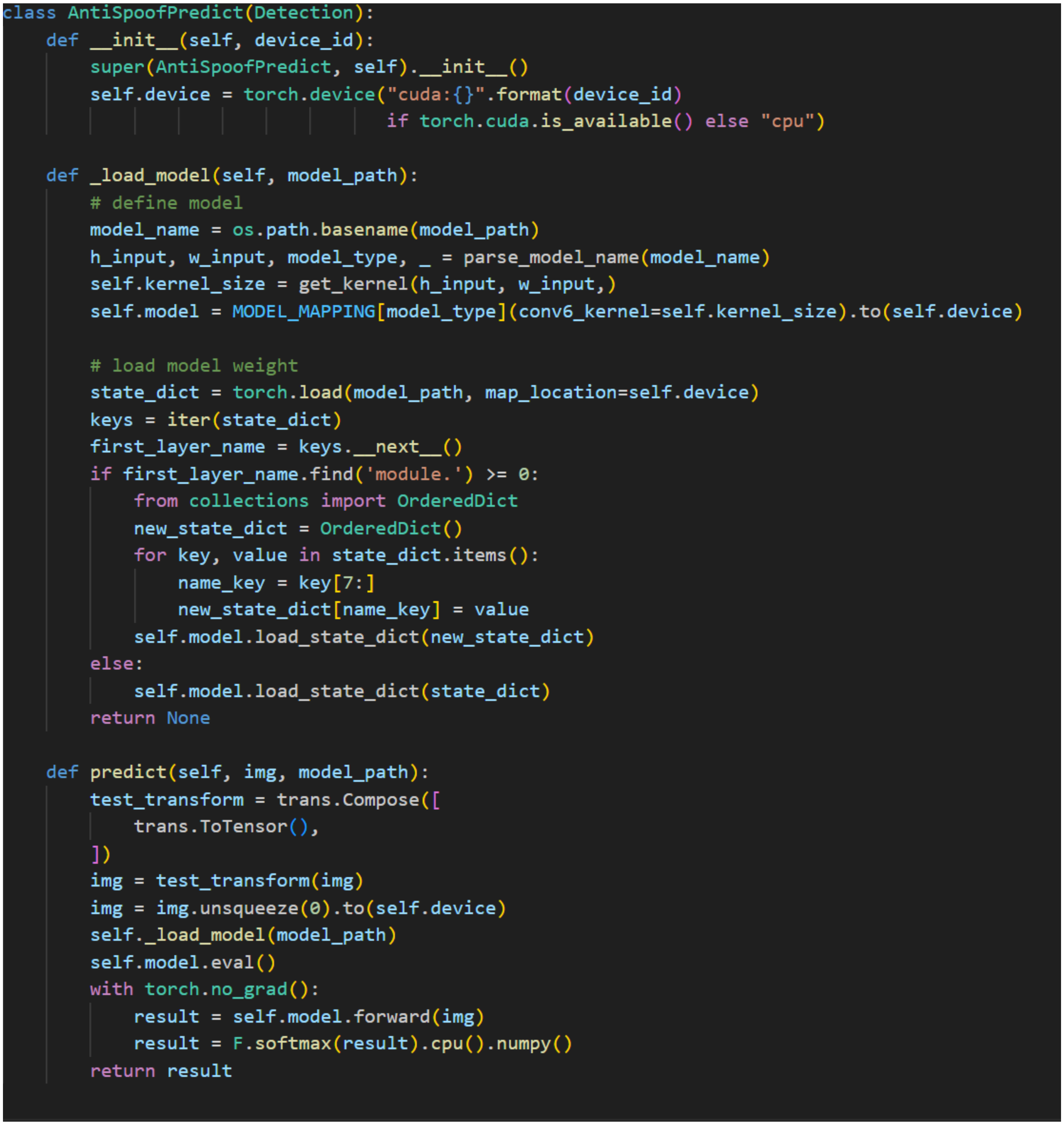
```python
def get_face_encodings(directory):
    face_encodings = []

    # Loop over all files in the directory
    for filename in os.listdir(directory):
        if filename.endswith(".jpg") or filename.endswith(".jpeg"):
            # Load the image file
            image = face_recognition.load_image_file(os.path.join(directory, filename))

            # Find face encodings for the image
            encodings = face_recognition.face_encodings(image)

            # If there are faces in the image
            if len(encodings) > 0:
                # Append the first face encoding
                face_encodings.append(encodings[0])
```

4.2 Recognizing each face and adding to excel file



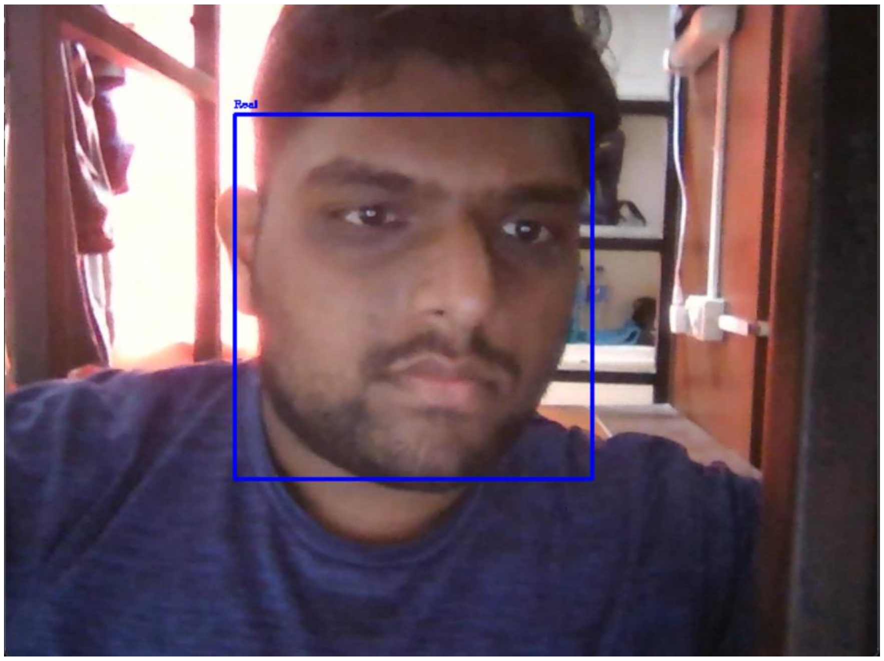
```python
def recognize_faces(image):
    # Load the known faces
    known_face_encodings = load_encodings('encodings.pkl')
    known_face_names = os.listdir("C:/Users/adity/OneDrive/Desktop/TARP Antispoof2/training_img")

    # Find all face locations and face encodings in the current frame
    face_locations = face_recognition.face_locations(image)
    face_encodings = face_recognition.face_encodings(image, face_locations)

    names = []
    for face_encoding in face_encodings:
        # See if this face is a match for any of our known faces
        matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
        name = "Unknown"

        # If a match was found, use the first one (order of known faces matters)
        if True in matches:
            first_match_index = matches.index(True)
            name = known_face_names[first_match_index]

        names.append(name)
```

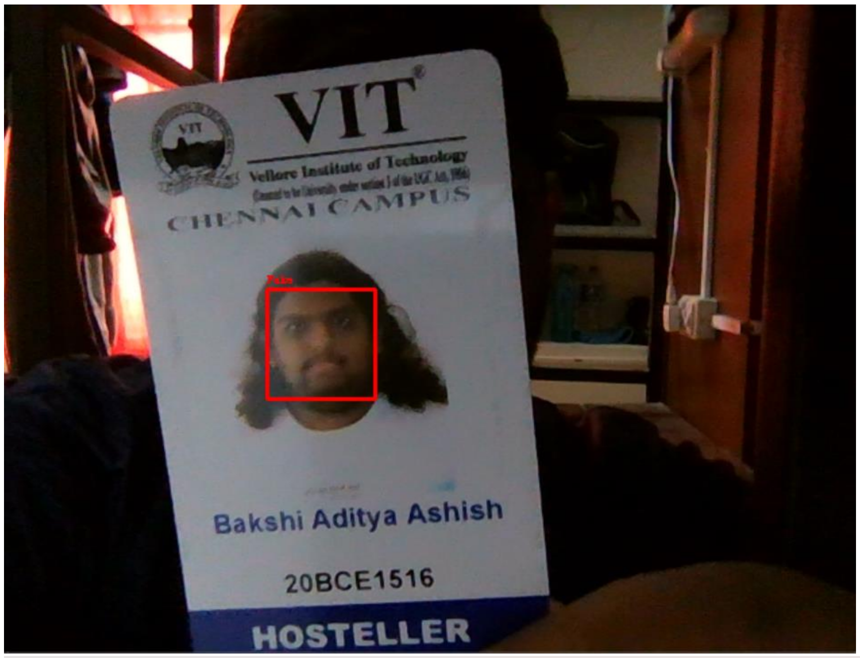
4.3 Detecting Liveness using trained CNN

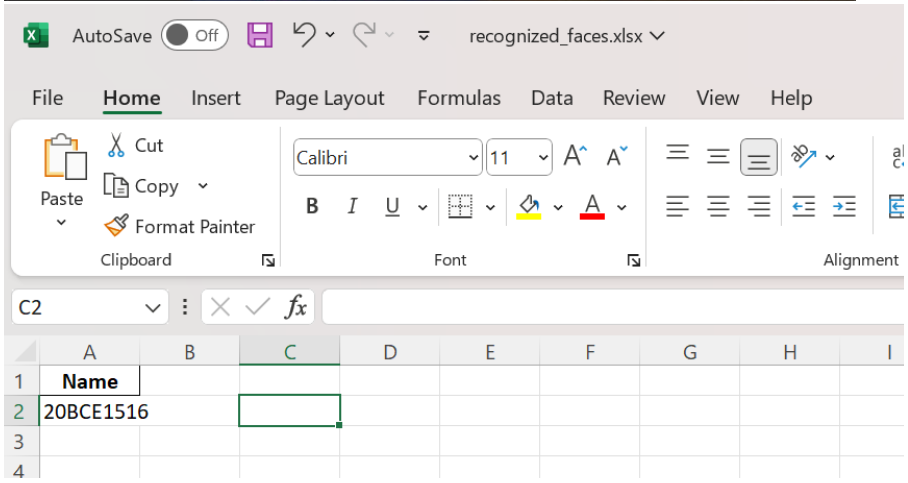


```python
class AntiSpoofPredict(Detection):
    def __init__(self, device_id):
        super(AntiSpoofPredict, self).__init__()
        self.device = torch.device("cuda:{}".format(device_id)
                                   if torch.cuda.is_available() else "cpu")

    def _load_model(self, model_path):
        # define model
        model_name = os.path.basename(model_path)
        h_input, w_input, model_type, _ = parse_model_name(model_name)
        self.kernel_size = get_kernel(h_input, w_input,)
        self.model = MODEL_MAPPING[model_type](conv6_kernel=self.kernel_size).to(self.device)

        # load model weight
        state_dict = torch.load(model_path, map_location=self.device)
        keys = iter(state_dict)
        first_layer_name = keys.__next__()
        if first_layer_name.find('module.') >= 0:
            from collections import OrderedDict
            new_state_dict = OrderedDict()
            for key, value in state_dict.items():
                name_key = key[7:]
                new_state_dict[name_key] = value
            self.model.load_state_dict(new_state_dict)
        else:
            self.model.load_state_dict(state_dict)
        return None

    def predict(self, img, model_path):
        test_transform = trans.Compose([
            trans.ToTensor(),
        ])
        img = test_transform(img)
        img = img.unsqueeze(0).to(self.device)
        self._load_model(model_path)
        self.model.eval()
        with torch.no_grad():
            result = self.model.forward(img)
            result = F.softmax(result).cpu().numpy()
        return result
```

Chapter 5: Results






AutoSave Off
recognized_faces.xlsx
File Home Insert Page Layout Formulas Data Review View Help
Paste Cut Copy Format Painter
Clipboard
Calibri 11
Font
Alignment
C2

	A
1	Name
2	20BCE1516

Chapter 6: Outcomes

1. **Automated Attendance System**: The successful completion of this project would result in an automated attendance system that uses face recognition and liveness detection technologies. This system could replace traditional attendance tracking methods, saving time and reducing errors.
2. **Improved Security**: The liveness detection component of the system would add an extra layer of security, preventing spoofing attacks and ensuring that the attendance system cannot be easily manipulated.
3. **User-Friendly Interface**: If designed well, the system could offer a user-friendly interface that requires minimal interaction from the users. Users would simply need to present their face to the camera to mark their attendance.
4. **Real-Time Processing**: The system would be capable of processing images in real-time, making it suitable for situations where immediate attendance tracking is required.
5. **Hygiene and Safety**: As a contactless method of attendance tracking, the system would be more hygienic than biometric systems that require physical contact. This could be particularly beneficial in the post-pandemic world.

Chapter 7: Single Unique Factor

- **Integration of Face Recognition and Liveness Detection**: While there are many projects that focus on either face recognition or liveness detection, this project integrates both. This not only enhances the functionality of the attendance system but also significantly improves its security.
- **Real-Time Processing**: The ability to process images in real-time is a notable feature. It allows immediate attendance tracking and can be particularly beneficial in situations where instant verification is required.
- **Contactless and Hygienic**: In the post-pandemic world, contactless technologies are preferred for hygiene reasons. This project leverages this trend, making it a safer alternative to biometric systems that require physical contact.
- **User-Friendly Interface**: The system is designed to be user-friendly, requiring minimal interaction from the users. Users simply need to present their face to the camera to mark their attendance, making it easy and convenient to use.

Chapter 8: Site Visit

During our stay in VIT, we observed the traditional attendance tracking methods in place and identified several areas where a face recognition and liveness detection attendance system could offer improvements.The current system, which relies on manual roll calls, was time-consuming and prone to errors. It also lacked the ability to prevent impersonation or proxy attendance. In contrast, a face recognition and liveness detection system could automate the attendance process, improve accuracy, and prevent spoofing attempts. However, implementing such a system would require addressing challenges such as varying lighting conditions, handling high volumes of individuals, and ensuring a user-friendly interface. This underscored the potential benefits of the proposed system, while also highlighting the practical considerations that need to be addressed for successful implementation.

Chapter 9: Sample Codes

GitHub Link: bakshi-aditya/TARP_JCOMP (github.com)

Chapter 10: References

1. “RetinaFace: Single-stage Dense Face Localisation in the Wild” deepinsight/insightface
2. “Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks” serengil/deepface
3. “Finding Tiny Faces” peiyunh/tiny
4. “Real-time Convolutional Neural Networks for Emotion and Gender Classification” oarriaga/face_classification
5. “LFFD: A Light and Fast Face Detector for Edge Devices” YonghaoHe/A-Light-and-Fast-Face-Detector-for-Edge-Devices
6. “BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs” PaddlePaddle/PaddleDetection
7. “FaceBoxes: A CPU Real-time Face Detector with High Accuracy” sfzhang15/FaceBoxes
8. “A Face Recognition Method in the Internet of Things for Security in Smart Recognition Places” Prof. Kalpana Malpe
9. “Hybrid HOG-SVM encrypted face detection and recognition model” Shilpa Sharma, Linesh Raja, Vaibhav Bhatnagar, Divya Sharma, Swami Nisha Bhagirath
10. “Face detection and recognition system based on hybrid statistical, machine learning and nature-based computing” M. Karnan, R. Vinodini