**Table of Contents**

1. Introduction

1.1 Background

1.2 Objectives

1. Methodology

2.1 System Design

2.2 Tools and Technologies

1. Implementation

3.1 Database Management (records.csv)

3.2 Helper Functions (utility\_fns.py)

3.3 Face Verification (app.py)

1. System Workflow

4.1 Input to Output Flow

4.2 User Interface Design

4.3 Design Considerations

1. Results and Discussion
2. Conclusion
3. References

**Face Verification System for Student Voting**

**1. Introduction**

**1.1 Background**

In the digital age, the integrity of online voting is crucial for educational institutions. Traditional identification methods, such as ID cards or passwords, are often prone to misuse or fraud. This project aims to develop a face verification system specifically designed for student voting at our institution. The system verifies a student's identity by comparing a webcam image with a stored image in the records, ensuring that the person casting the vote is indeed the registered student.

**1.2 Objectives**

* To develop a secure and efficient face verification system.
* To integrate a record folder for storing student information and images.
* To implement a user-friendly interface for data entry and verification.
* To utilize advanced face recognition techniques for accurate verification.

**2. Methodology**

**2.1 System Design**

The system comprises several components: a front-end interface for student interaction, a backend for database management, and a face verification module using DeepFace. The process flow includes:

1. **User Input**: Students enter their Full name and matriculation number in the format XXX/YY/ZZZZ.
2. **Image Capture**: The system captures the student's image via a webcam.
3. **Database Retrieval**: The system scans through the records to retrieve the corresponding student's data and image from the database.
4. **Verification**: The captured image is compared with the retrieved image using face verification algorithms.

**2.2 Tools and Technologies**

* **Programming Language**: Python
* **Frameworks**: Gradio for the user interface, pandas for keeping records.
* **Libraries**: DeepFace for face verification, OpenCV for image processing, PIL for image loading and handling, and matplotlib for image loading.

**3. Implementation**

**3.1 User Interface (app.py)**

The user interface is built using Gradio, a library for creating web interfaces for machine learning models. Gradio provides an interactive platform where users can enter data and view results. The interface includes:

* **Textboxes**: For entering the last name and matric number.
* **Image Capture**: A component to capture images from the webcam.
* **Display Areas**: To show the captured image and the image retrieved from the database.
* **Verify Button**: To initiate the verification process.

**Technical Terms Explained**

* **Gradio**: A Python library that simplifies the process of creating user interfaces for machine learning applications. It allows developers to easily build web-based GUIs to interact with their models, providing components like input boxes, image display areas, and buttons.

**3.2 Records(records.csv)**

The records are implemented using pandas. The records store the following details:

* **Matric Number**
* **Image**: Path to the corresponding student images in the record folder.

**3.3 Helper Functions (utility\_fns.py)**

Several utility functions are used throughout the project:

* **make\_records**: Scans through the repository containing student information(images) and creates records that are easily accessible for verification
* **empty\_img:** Provides an empty image in array format to be visualized whenever a match in the records is not found.

**Technical Terms Explained**

* **Image Cropping**: The process of removing the outer parts of an image to focus on a specific area, such as a face. This is important for improving the accuracy of facial recognition by focusing on the relevant features.

**3.4 Face Verification (app.py)**

The core of the face verification process uses the DeepFace library, a Python framework for deep learning-based facial recognition and analysis. The project employs the following components:

1. **DeepFace Framework**: DeepFace is an open-source library that provides a standardized interface for various deep learning-based face recognition models. It supports multiple models, allowing for flexible and customizable face analysis.
2. **Facenet Model**: Facenet is a deep learning model designed to create a compact and discriminative embedding for face images. It maps face images to a Euclidean space where the distance between two points indicates the similarity between two faces. More details can be found in the [original paper](https://arxiv.org/abs/1503.03832).
3. **RetinaFace Detector**: RetinaFace is a state-of-the-art face detection model that identifies facial landmarks such as eyes, nose, and mouth. It helps in accurately detecting and aligning faces in images, which is crucial for achieving high verification accuracy. More information can be found in the [original paper](https://arxiv.org/abs/1905.00641).
4. **Cosine Similarity**: A metric used to measure the cosine of the angle between two non-zero vectors in an inner product space. In this project, it measures the similarity between the embeddings of two face images. The cosine similarity ranges from -1 to 1, with 1 indicating identical vectors (i.e., identical faces).
5. **Normalization**: This refers to the process of adjusting the pixel values of an image to a standard range or distribution. In the context of Facenet, normalization ensures that the input images are preprocessed in a way that matches the data used to train the model, which improves the accuracy of the generated embeddings.

**4. System Workflow**

**4.1 Input to Output Flow**

The application workflow is designed to provide a seamless experience from input to output:

1. **Input Phase**:
   * The student is prompted to enter their last name and matriculation number into the text fields provided in the Gradio interface.
   * The student then proceeds to capture their image using the webcam.
2. **Processing Phase**:
   * Once the submit button is clicked, the system retrieves the student's record from the database using the entered details (last name and matriculation number).
   * The stored image associated with the student is fetched from the database.
   * The captured image from the webcam and the retrieved image from the database are both processed and prepared for comparison. This involves detecting and aligning faces using RetinaFace and then normalizing the images.
3. **Verification Phase**:
   * The DeepFace library, with the Facenet model, generates embeddings for both images. These embeddings are numerical representations of the faces, capturing unique facial features.
   * The system calculates the cosine similarity between the two embeddings to assess how closely the images match.
   * A similarity score is generated, and based on a predefined threshold, the system determines whether the verification is successful.
4. **Output Phase**:
   * The result is displayed on the interface, indicating whether the student's identity has been verified successfully.

**4.2 User Interface Design**

The Gradio interface is designed with simplicity and usability in mind:

* **Clear Input Fields**: The textboxes for entering the last name and matric number are clearly labeled, minimizing the chances of user error.
* **Real-Time Feedback**: The webcam component provides real-time feedback, allowing students to adjust their position before capturing the image.
* **Visual Confirmation**: The interface displays both the captured image and the retrieved image, providing visual confirmation of the process.
* **Result Display**: The result is displayed prominently, ensuring that the outcome of the verification is clear to the user.

**4.3 Design Considerations**

Several design considerations were taken into account:

* **Accessibility**: The interface is simple and intuitive, ensuring that users of all technical backgrounds can use the system without difficulty.
* **Security**: The system ensures that sensitive data, such as images and personal information, is handled securely, adhering to data protection standards.
* **Scalability**: The design allows for easy scaling, accommodating a growing number of students and increasing the database's size without performance degradation.
* **Performance**: The choice of lightweight technologies like SQLite and optimized image processing techniques ensures that the system operates efficiently, providing quick responses during the verification process.

**5. Results and Discussion**

The system was tested with a dataset of student images. The face verification process showed high accuracy in matching the correct student with the stored image, demonstrating the effectiveness of using DeepFace with the specified model and settings. The choice of Facenet and RetinaFace provided robust performance, handling various lighting conditions and facial orientations.

Challenges encountered included:

* **Lighting Variations**: Inconsistent lighting affected the quality of images captured from the webcam.
* **Resolution and Quality**: Differences in image resolution and quality between the stored images and webcam captures impacted the verification accuracy.

To mitigate these issues, preprocessing steps like image normalization and histogram equalization were considered.

**6. Conclusion**

The face verification system successfully meets the project's objectives, providing a secure and efficient way to verify student identities for computer-based tests. The combination of DeepFace, Facenet, and RetinaFace proved effective in handling the verification process. Future work could include enhancing the system's robustness to different environmental conditions, such as varying lighting and background settings, and extending its application to other areas requiring identity verification.

**7. References**

* [DeepFace GitHub Repository](https://github.com/serengil/deepface)
* [Facenet: A Unified Embedding for Face Recognition and Clustering](https://arxiv.org/abs/1503.03832)
* [RetinaFace: Single-shot Multi-level Face Localisation in the Wild](https://arxiv.org/abs/1905.00641)
* [Gradio: Build and share delightful machine learning apps](https://www.gradio.app/)