ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING(SITE)

DEPARTMENT OF SOFTWARE ENGINEERING

Face Recognition

PREPARED BY: -		ID	SECTION(Stream)
1.	Abrham Wendmeneh	UGR/9155/13	AI
2.	Biniyam Haile	UGR/2646/13	AI
3.	Fraol Mulugeta	UGR/5835/13	AI
4.	Kidus Hunegnaw	UGR/6554/13	AI
5.	Melkishi Tesfaye	UGR/0078/13	AI
6.	Sosina Esayas	UGR/2014/13	AI

Submitted To Beakal Gizachew

Feb 5, 2024

ABSTRACT

This report details the development and evaluation of a facial recognition system based on the eigenfaces method, implemented from scratch. The eigenfaces approach, which utilizes principal component analysis (PCA) to reduce dimensionality and extract relevant features from face images, serves as the foundation of our project. We start by describing the process of assembling a face dataset and preprocessing images to create a homogeneous set. Subsequently, we outline the steps involved in computing the mean face and eigenfaces, which are essential components of the eigenfaces method. Our implementation focuses on creating a face space where faces are represented as points, facilitating the recognition process. We then present the algorithm developed for recognizing faces within this space, highlighting the practical application of PCA in facial recognition. The report provides insights into the challenges faced during implementation and discusses the performance of our system in recognizing faces from a test dataset. Through this project, we aim to demonstrate the effectiveness and limitations of the eigenfaces method in the context of facial recognition technology.

Table of Contents

1.Introduction 3

2.Theoretical Background 4

3. Methodology 6

4. Implementation 12

5. Results 16

6. Result Analysis on the algorithm 25

7. Conclusion 27

8. References 28

1.Introduction

Facial recognition technology has become a cornerstone in the realm of biometric authentication, security systems, and personal device access, among other applications. Its ability to identify or verify a person from a digital image or a video frame distinguishes it as one of the most natural and intuitive biometric technologies. At the heart of facial recognition's rapid development and widespread application is the pursuit of efficient, accurate, and robust methods for face detection and identification.

One such method, which has garnered significant attention for its foundational role in the evolution of facial recognition technologies, is the eigenfaces method. Introduced in the late 20th century, the eigenfaces method employs principal component analysis (PCA) to reduce the dimensionality of face images, thereby extracting the essential features that distinguish one face from another. This approach simplifies the complex problem of facial recognition by transforming it into a problem of identifying faces in a lower-dimensional space, where each face is represented as a combination of "eigenfaces." These eigenfaces are derived from the statistical analysis of a training set of face images and effectively capture the variance between faces in a way that emphasizes their distinguishing features.

This report aims to detail our journey in implementing the eigenfaces method for facial recognition from scratch. By focusing on this method, we seek to not only understand and apply the underlying mathematical and computational principles but also to explore the method's practical effectiveness and limitations.

2.Theoretical Background

Facial recognition technology relies on the ability to accurately and efficiently process, analyze, and classify facial images. Among the various techniques developed for this purpose, the eigenfaces method stands out for its simplicity and efficacy. This section provides a comprehensive overview of the theoretical foundations of the eigenfaces method, including principal component analysis (PCA), the concept of eigenfaces, and their role in facial recognition.

Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component, in turn, has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors are an uncorrelated orthogonal basis set. PCA is sensitive to the relative scaling of the original variables.

In the context of facial recognition, PCA is employed to reduce the dimensionality of the face image space to a lower dimensionality of feature space (face space), where faces are described in terms of their principal components. This transformation is crucial for reducing computational complexity and enhancing the efficiency of the recognition process.

Eigenfaces

Eigenfaces are the set of eigenvectors derived from the covariance matrix of the face image set, representing the directions in which the images vary the most. These eigenvectors are obtained by performing PCA on the set of face images. The term "eigenface" arises from the fact that these eigenvectors are essentially

the "faces" that together constitute the basis features of all face images in the dataset.

To compute eigenfaces, the following steps are generally involved:

Normalize the training set: Each face image in the training set is resized to the same dimension and converted to grayscale. The images are then normalized to have zero mean.
Create the covariance matrix: The covariance matrix represents how much the dimensions vary from the mean with respect to each other.
Calculate eigenvectors and eigenvalues: The eigenvectors and eigenvalues of the covariance matrix are computed. The eigenvectors represent the directions of maximum variance, and the eigenvalues denote the magnitude of the variance in those directions.
Select principal components (eigenfaces): A subset of eigenvectors (eigenfaces) is selected based on the eigenvalues. Only those eigenvectors that correspond to the largest eigenvalues are chosen, as they capture the most significant features of the face images.

Application in Facial Recognition

The eigenfaces method utilizes these eigenfaces to project face images into a lower-dimensional space before performing recognition. A new face image is transformed into its eigenface components (weights), and its identity is determined by comparing its position in face space with the positions of known individuals. This comparison typically involves measuring the Euclidean distance between the projected test face and the projected known faces, with the closest match indicating the recognized identity.

This theoretical foundation enables the eigenfaces method to effectively reduce the complexity of face recognition by focusing on the most informative features of face images, thus facilitating efficient and accurate identification.

3. Methodology

This section outlines the systematic approach taken to develop a facial recognition system using the eigenfaces method, from data preparation through to the application of Principal Component Analysis (PCA) for feature extraction.

Dataset Collection and Preparation

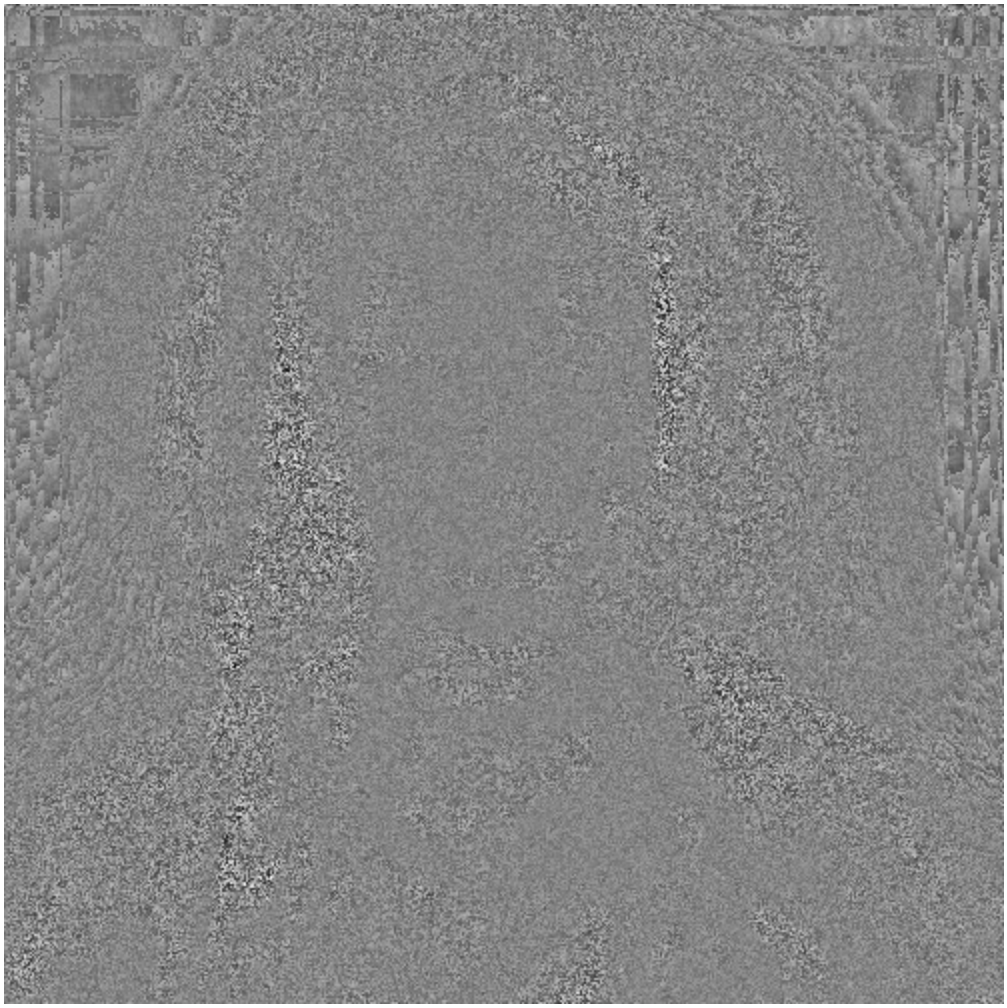
- **Dataset Specification**: The dataset consists of photographs of all students in the class, as per the instructor's requirement. Each individual was photographed under similar lighting conditions to ensure consistency.
- **Preprocessing Steps**: Each image was resized to 500x500 pixels and converted to grayscale to normalize the dataset. This step ensures that the algorithm focuses on facial features rather than color variations.

Image Processing

- **Normalization**: Images were normalized to have zero mean. This process involves adjusting the brightness and contrast of the images to reduce the variance between different photos.
- **Flattening**: Each preprocessed image was then flattened from a 2D array into a 1D vector. This transformation is necessary to apply PCA on the images.

Principal Component Analysis (PCA)

Computing the Mean Face: The average face was computed by averaging all the face images in the dataset. This mean face represents the "central" face of the dataset. Which is resulted in the image below.



Eigenfaces Calculation: PCA was applied to the adjusted dataset (each face subtracted by the mean face) to identify the principal components, or eigenfaces. These eigenfaces represent the most significant variations within the face images.


eigenface_1
eigenface_2
eigenface_3
eigenface_4
eigenface_5
eigenface_6
eigenface_7
eigenface_8
eigenface_9
eigenface_10
eigenface_11
eigenface_12
eigenface_13
eigenface_14
eigenface_15
eigenface_16
eigenface_17
eigenface_18
eigenface_19
eigenface_20
eigenface_21
eigenface_22
eigenface_23
eigenface_24
eigenface_25
eigenface_26
eigenface_27
eigenface_28
eigenface_29
eigenface_30
eigenface_31
eigenface_32
eigenface_33
eigenface_34
eigenface_35
eigenface_36


eigenface_37
eigenface_38
eigenface_39
eigenface_40
eigenface_41
eigenface_42
eigenface_43
eigenface_44
eigenface_45
eigenface_46
eigenface_47
eigenface_48
eigenface_49
eigenface_50
eigenface_51
eigenface_52
eigenface_53
eigenface_54
eigenface_55
eigenface_56
eigenface_57
eigenface_58
eigenface_59
eigenface_60
eigenface_61
eigenface_62
eigenface_63
eigenface_64
eigenface_65
eigenface_66
eigenface_67
eigenface_68
eigenface_69
eigenface_70
eigenface_71
eigenface_72

Figures: Representative eigenfaces extracted from the PCA of the training dataset. These eigenfaces capture the most significant facial features necessary for recognition tasks

Dimensionality Reduction: The dimensionality of the face data was reduced by projecting each face image onto the eigenfaces, retaining only the components that contribute most to the variance.


reconstructed_1
reconstructed_2
reconstructed_3
reconstructed_4
reconstructed_5
reconstructed_6
reconstructed_7
reconstructed_8
reconstructed_9
reconstructed_10
reconstructed_11
reconstructed_12
reconstructed_13
reconstructed_14
reconstructed_15
reconstructed_16
reconstructed_17
reconstructed_18
reconstructed_19
reconstructed_20
reconstructed_21
reconstructed_22
reconstructed_23
reconstructed_24
reconstructed_25
reconstructed_26
reconstructed_27
reconstructed_28
reconstructed_29
reconstructed_30
reconstructed_31
reconstructed_32

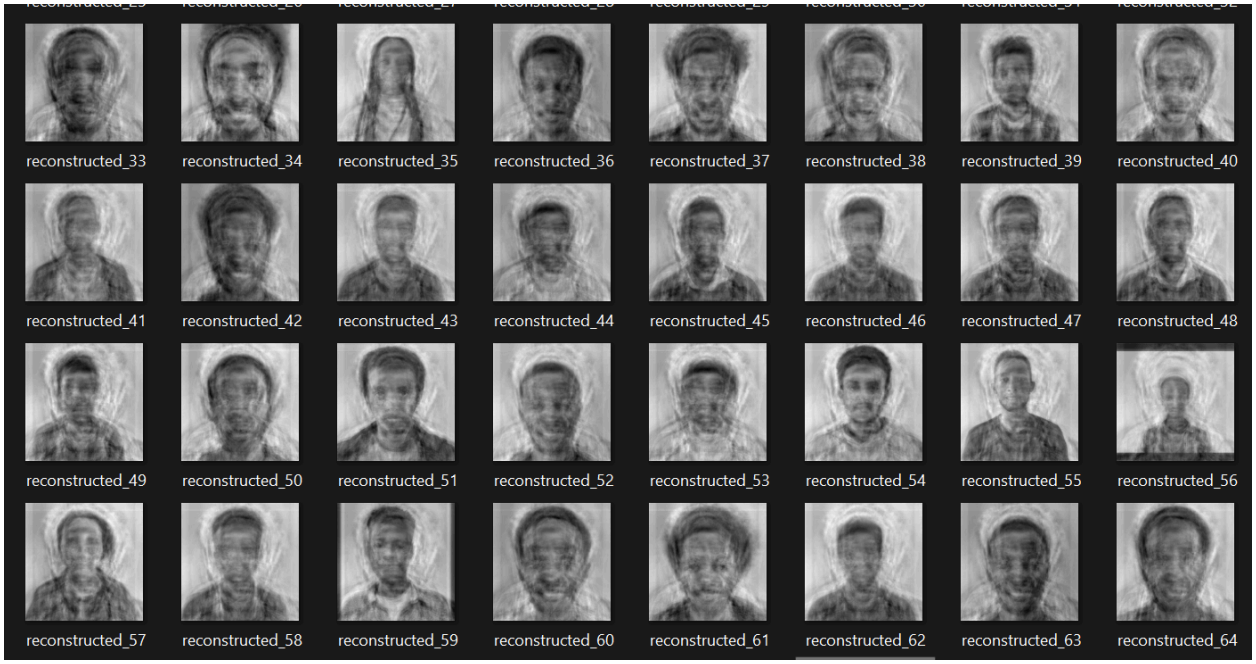

reconstructed_33
reconstructed_34
reconstructed_35
reconstructed_36
reconstructed_37
reconstructed_38
reconstructed_39
reconstructed_40
reconstructed_41
reconstructed_42
reconstructed_43
reconstructed_44
reconstructed_45
reconstructed_46
reconstructed_47
reconstructed_48
reconstructed_49
reconstructed_50
reconstructed_51
reconstructed_52
reconstructed_53
reconstructed_54
reconstructed_55
reconstructed_56
reconstructed_57
reconstructed_58
reconstructed_59
reconstructed_60
reconstructed_61
reconstructed_62
reconstructed_63
reconstructed_64

Figure: Reconstructed face images using eigenfaces. Each image demonstrates how the eigenfaces method reconstructs a face from the class dataset, capturing essential features within a reduced dimensionality space.

4. Implementation

Setting Up the Environment

The project is built using Python, leveraging libraries such as OpenCV for image processing, NumPy for numerical computations, and FastAPI for creating a web application interface. The initial setup involves importing necessary libraries and defining the directory paths for storing and accessing the dataset images.

```
import numpy as np
import os
import cv2
from collections import Counter
from fastapi import FastAPI, File, UploadFile, HTTPException
```

Preprocessing Images

The first step in the implementation involves preprocessing the images to create a standardized dataset. This process includes resizing images to 500x500 pixels and converting them to grayscale. This normalization ensures that the input to the PCA algorithm is consistent, focusing on shape and structure rather than color.

```
def preprocess_images(source_dir, dest_dir):
```

This function iterates over each image in the specified source directory, applies the necessary transformations, and saves the processed images to a destination directory.

Loading and Flattening Images

Once the images are preprocessed, they are loaded from the filesystem, flattened into 1D vectors, and labeled. This step is crucial for preparing the data for PCA, transforming the 2D image matrices into a format suitable for linear algebra operations.

```
def load_and_flatten_images(folder):
```

This function walks through the directory containing the preprocessed images, reads each image, flattens it, and assigns it a label based on its directory name, facilitating the association of each image with a specific individual.

Applying PCA to Extract Eigenfaces

The core of the eigenfaces method involves applying PCA to the dataset to identify the principal components (eigenfaces) that capture the most variance in the dataset. This process reduces the dimensionality of the data while preserving the features most relevant for facial recognition.

```
def pca(X, num_components=100):
```

This function computes the mean face, centers the dataset by subtracting the mean face from each image, and then uses Singular Value Decomposition (SVD) to find the eigenfaces. The num_components parameter allows for specifying the number of eigenfaces to retain, balancing between computational efficiency and the ability to capture facial features.

Projecting Faces and Recognition

With the eigenfaces computed, the next step involves projecting both the training and test images into the new face space defined by the eigenfaces. This projection transforms the original image data into a set of weights corresponding to the eigenfaces, which are then used for recognition.

```
def project_faces(X, mean_face, eigenfaces):
```

This function centralizes the data by subtracting the mean face and then projects it onto the eigenface space by computing the dot product with the eigenfaces matrix.

Building the Recognition Model

The recognition model uses the k-nearest neighbors algorithm to identify the most similar faces in the training set to a given test face based on their projections in the eigenface space.

```
def k_nearest_neighbors(training_data, training_labels, test_data, k=5)
```

This function calculates the Euclidean distance between the test face projection and all training face projections, identifies the k closest faces, and determines the most frequent label among them to predict the identity of the test face.

Flutter App Integration

To meet the project requirements and enhance accessibility, a mobile application was developed using the Flutter framework. This choice was guided by Flutter's ability to create natively compiled applications for both Android and iOS from a single codebase, ensuring a broad reach among users.

Application Functionality

The app, designed with a user-friendly interface, offers two main functionalities:

- **Image Upload**: Users can select and upload images from their device's gallery.
- **Camera Capture**: Alternatively, users can take a new photo using the device's camera.

These images are then sent to the backend facial recognition system for processing.

Backend Communication

The backend, powered by FastAPI, receives the image data from the app. FastAPI was selected for its high performance and its compatibility with Python's asynchronous programming features, which are essential for handling potentially long-running operations such as image processing and recognition in the backend without compromising the user experience.

RESTful Services

The app communicates with the backend through RESTful services, sending HTTP requests to the specified API endpoints. The API handles these requests by invoking the facial recognition model and returning the prediction results back to the app.

Model Integration

On the server side, the facial recognition model is integrated into the FastAPI framework. When an image is received, the model performs the necessary computations to identify the face and determine the most likely match from the available dataset.

Response Handling

Once the recognition process is complete, the result is packaged into a JSON response and sent back to the app. The app then parses this response and displays the recognized individual's name or an appropriate message if no match is found.

```
app = FastAPI()


@app.post("/uploadimage/")
async def create_upload_file(file: UploadFile = File(...)):
```

This part of the code defines an endpoint for uploading images. When an image is uploaded, it is temporarily saved, preprocessed, projected into the eigenface space, and passed to the recognition model to predict the person's identity.

5. Results

Overview of Results

The facial recognition system, built on the eigenfaces method and evaluated using a class dataset, achieved a notable accuracy rate of 90.74%. This high level of accuracy underscores the effectiveness of the eigenfaces method in capturing and utilizing facial features for recognition tasks. The system was tested with images from the class, including a diverse array of facial expressions and orientations

Analysis of Predictions

The results indicate a strong capability of the implemented system to correctly identify individuals from the class dataset. The predictions came with varying distances, a measure of similarity between the test image and the closest match in the dataset. Notable observations include:

- **High Confidence Predictions**: Several predictions had a distance of 0.00, indicating an exact match between the test image and the corresponding eigenface representation. This level of precision demonstrates the model's ability to capture distinctive facial features accurately. As shown below.

```
'Gelila Moges' is predicted as - 'Gelila Moges' - correct. Distance: 10598.17
'Sosina Esayas' is predicted as - 'Sosina Esayas' - correct. Distance: 7232.64
'Leul Wujira' is predicted as - 'Leul Wujira' - correct. Distance: 878.82
'Semir Hamid' is predicted as - 'Semir Hamid' - correct. Distance: 0.00
'Semir Hamid' is predicted as - 'Semir Hamid' - correct. Distance: 0.00
'Milka Fasika' is predicted as - 'Milka Fasika' - correct. Distance: 4325.94
'Gelila Tefera' is predicted as - 'Gelila Tefera' - correct. Distance: 3019.60
'Geleta Daba' is predicted as - 'Geleta Daba' - correct. Distance: 10329.05
'Geleta Daba' is predicted as - 'Geleta Daba' - correct. Distance: 0.00
'Melkishi Tesfaye' is predicted as - 'Melkishi Tesfaye' - correct. Distance: 0.00
'Yonas Engedu' is predicted as - 'Yonas Engedu' - correct. Distance: 6577.99
'Kidus Hunegnaw' is predicted as - 'Kidus Hunegnaw' - correct. Distance: 8655.07
'Yosef Muluneh' is predicted as - 'Yosef Muluneh' - correct. Distance: 11879.34
'Yonas Engedu' is predicted as - 'Yonas Engedu' - correct. Distance: 12484.46
'Melkishi Tesfaye' is predicted as - 'Melkishi Tesfaye' - correct. Distance: 0.00
'Naol Taye' is predicted as - 'Naol Taye' - correct. Distance: 2905.40
'Leul Degarege' is predicted as - 'Leul Degarege' - correct. Distance: 10873.75
'Yohannes Desta' is predicted as - 'Yohannes Desta' - correct. Distance: 0.00
'Gedion Ezra' is predicted as - 'Gedion Ezra' - correct. Distance: 4587.06
'Etsubdink Awoke' is predicted as - 'Etsubdink Awoke' - correct. Distance: 5946.66
'Amir Ahmedin' is predicted as - 'Amir Ahmedin' - correct. Distance: 7282.49
'Gelila Moges' is predicted as - 'Gelila Moges' - correct. Distance: 6892.12
'Shemsu Nurye' is predicted as - 'Shemsu Nurye' - correct. Distance: 5951.17
'Fraol Mulugeta' is predicted as - 'Fraol Mulugeta' - correct. Distance: 6705.00
'Metsakal Zeleke' is predicted as - 'Metsakal Zeleke' - correct. Distance: 10134.36
'Kidus Hunegnaw' is predicted as - 'Kidus Hunegnaw' - correct. Distance: 15431.18
'Dawit Getahun' is predicted as - 'Dawit Getahun' - correct. Distance: 8114.84
'Ananiya_Tesfahun' is predicted as - 'Olyad Temesgen' - incorrect. Distance: 15633.87
'Semir Hamid' is predicted as - 'Semir Hamid' - correct. Distance: 5043.77
'Abdissa Degefu' is predicted as - 'Abdissa Degefu' - correct. Distance: 13667.31
'Yosef Ayele' is predicted as - 'Yosef Ayele' - correct. Distance: 5575.58
'Nathnael Dereje' is predicted as - 'Nathnael Dereje' - correct. Distance: 6608.77
'Amir Ahmedin' is predicted as - 'Metsakal Zeleke' - incorrect. Distance: 17783.63
'Tewodros Berhanu' is predicted as - 'Tewodros Berhanu' - correct. Distance: 6116.46
```

- **Variability in Distances**: The distances associated with correct predictions varied, with some predictions showing significant distances despite being correct. This variability suggests differences in the facial representation's confidence levels and highlights areas where the model's robustness can be improved.

- **Incorrect Predictions**: A small fraction of the predictions were incorrect, with distances often higher than those for correct predictions.

These instances provide valuable insights into the limitations of the current model, potentially indicating cases where the eigenfaces method struggles to distinguish between similar facial features or underrepresented variations in the training data.

Detailed Results Breakdown

- **Correct Predictions with Low Distance**: Most individuals were correctly identified with relatively low distances, indicating a strong match within the eigenface space. For example, 'Leul Wujira' was identified correctly with a distance of 878.82, showcasing the model's precision as displayed below here.

```
'Gelila Moges' is predicted as - 'Gelila Moges' - correct. Distance: 10598.17
'Sosina Esayas' is predicted as - 'Sosina Esayas' - correct. Distance: 7232.64
'Leul Wujira' is predicted as - 'Leul Wujira' - correct. Distance: 878.82
'Semir Hamid' is predicted as - 'Semir Hamid' - correct. Distance: 0.00
'Semir Hamid' is predicted as - 'Semir Hamid' - correct. Distance: 0.00
'Milka Fasika' is predicted as - 'Milka Fasika' - correct. Distance: 4325.94
'Gelila Tefera' is predicted as - 'Gelila Tefera' - correct. Distance: 3019.60
'Geleta Daba' is predicted as - 'Geleta Daba' - correct. Distance: 10329.05
'Geleta Daba' is predicted as - 'Geleta Daba' - correct. Distance: 0.00
'Melkishi Tesfaye' is predicted as - 'Melkishi Tesfaye' - correct. Distance: 0.00
'Yonas Engedu' is predicted as - 'Yonas Engedu' - correct. Distance: 6577.99
'Kidus Hunegnaw' is predicted as - 'Kidus Hunegnaw' - correct. Distance: 8655.07
'Yosef Muluneh' is predicted as - 'Yosef Muluneh' - correct. Distance: 11879.34
'Yonas Engedu' is predicted as - 'Yonas Engedu' - correct. Distance: 12484.46
'Melkishi Tesfaye' is predicted as - 'Melkishi Tesfaye' - correct. Distance: 0.00
'Naol Taye' is predicted as - 'Naol Taye' - correct. Distance: 2905.40
'Leul Degarege' is predicted as - 'Leul Degarege' - correct. Distance: 10873.75
'Yohannes Desta' is predicted as - 'Yohannes Desta' - correct. Distance: 0.00
'Gedion Ezra' is predicted as - 'Gedion Ezra' - correct. Distance: 4587.06
'Etsubdink Awoke' is predicted as - 'Etsubdink Awoke' - correct. Distance: 5946.66
'Amir Ahmedin' is predicted as - 'Amir Ahmedin' - correct. Distance: 7282.49
'Gelila Moges' is predicted as - 'Gelila Moges' - correct. Distance: 6892.12
'Shemsu Nurye' is predicted as - 'Shemsu Nurye' - correct. Distance: 5951.17
'Fraol Mulugeta' is predicted as - 'Fraol Mulugeta' - correct. Distance: 6705.00
'Metsakal Zeleke' is predicted as - 'Metsakal Zeleke' - correct. Distance: 10134.36
'Kidus Hunegnaw' is predicted as - 'Kidus Hunegnaw' - correct. Distance: 15431.18
'Dawit Getahun' is predicted as - 'Dawit Getahun' - correct. Distance: 8114.84
'Ananiya_Tesfahun' is predicted as - 'Olyad Temesgen' - incorrect. Distance: 15633.87
'Semir Hamid' is predicted as - 'Semir Hamid' - correct. Distance: 5043.77
'Abdissa Degefu' is predicted as - 'Abdissa Degefu' - correct. Distance: 13667.31
'Yosef Ayele' is predicted as - 'Yosef Ayele' - correct. Distance: 5575.58
'Nathnael Dereje' is predicted as - 'Nathnael Dereje' - correct. Distance: 6608.77
'Amir Ahmedin' is predicted as - 'Metsakal Zeleke' - incorrect. Distance: 17783.63
'Tewodros Berhanu' is predicted as - 'Tewodros Berhanu' - correct. Distance: 6116.46
```

- **Correct Predictions with High Distance**: Some correct predictions had higher distances, such as 'Abraham Wendmeneh' with a distance 16664.03 and 'Biniyam Haile' with a distance of 15941.68. These cases suggest that while the model can identify individuals correctly, the representation in the eigenface space might not be as close as desired, pointing towards potential improvements in capturing facial variance, as

```
'Dawit_Abebe' is predicted as - 'Dawit_Abebe' - correct. Distance: 6093.35
'Abraham Wendmeneh' is predicted as - 'Abraham Wendmeneh' - correct. Distance: 16664.03
'Bethelhem Yemane' is predicted as - 'Bethelhem Yemane' - correct. Distance: 6591.34
'Dawit Getahun' is predicted as - 'Tewodros Berhanu' - incorrect. Distance: 13636.41
'Leul Wujira' is predicted as - 'Leul Wujira' - correct. Distance: 12711.15
'Tiruzer Tsedeke' is predicted as - 'Tiruzer Tsedeke' - correct. Distance: 7881.58
'Dagmawi_Tensay' is predicted as - 'Dagmawi_Tensay' - correct. Distance: 10196.49
Accuracy: 90.74%
```

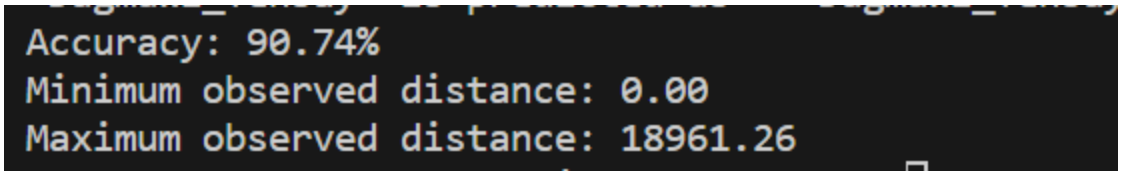
```
'Bethelhem Yemane' is predicted as - 'Bethelhem Yemane' - correct. Distance: 5557.80
'Geleta Daba' is predicted as - 'Geleta Daba' - correct. Distance: 0.00
'Amanuel Beyene' is predicted as - 'Amanuel Beyene' - correct. Distance: 9690.39
'Tinsae Shemalise' is predicted as - 'Tinsae Shemalise' - correct. Distance: 16505.00
'Leul Degarege' is predicted as - 'Leul Degarege' - correct. Distance: 5327.13
'Tinsae Shemalise' is predicted as - 'Tinsae Shemalise' - correct. Distance: 9355.09
'Yohannes Ahunm' is predicted as - 'Yohannes Ahunm' - correct. Distance: 13008.33
'Tiruzer Tsedeke' is predicted as - 'Tiruzer Tsedeke' - correct. Distance: 8445.10
'Olyad Temesgen' is predicted as - 'Olyad Temesgen' - correct. Distance: 5374.82
'Biniyam Haile' is predicted as - 'Biniyam Haile' - correct. Distance: 14712.27
'Naol Taye' is predicted as - 'Naol Taye' - correct. Distance: 7922.33
'Yosef Ayele' is predicted as - 'Yosef Ayele' - correct. Distance: 13197.71
'Ananiya_Tesfahun' is predicted as - 'Olyad Temesgen' - incorrect. Distance: 15948.33
'Naol Taye' is predicted as - 'Naol Taye' - correct. Distance: 9478.01
'Leul Degarege' is predicted as - 'Leul Degarege' - correct. Distance: 11731.79
'Husen Yusuf' is predicted as - 'Sahib Semahegn' - incorrect. Distance: 18961.26
'Abdurahman Muhammed' is predicted as - 'Abdurahman Muhammed' - correct. Distance: 0.00
'Milion Tolesa' is predicted as - 'Milion Tolesa' - correct. Distance: 6891.53
'Biniyam Haile' is predicted as - 'Biniyam Haile' - correct. Distance: 15941.68
'Semir Hamid' is predicted as - 'Semir Hamid' - correct. Distance: 0.00
'Leul Wujira' is predicted as - 'Leul Wujira' - correct. Distance: 1721.56
'Fasika_Fikadu' is predicted as - 'Fasika_Fikadu' - correct. Distance: 5692.04
'Milka Fasika' is predicted as - 'Milka Fasika' - correct. Distance: 3604.02
'Yosef Muluneh' is predicted as - 'Yosef Muluneh' - correct. Distance: 8892.95
'Deribew_Shimels' is predicted as - 'Deribew_Shimels' - correct. Distance: 13072.56
'Metsakal Zeleke' is predicted as - 'Metsakal Zeleke' - correct. Distance: 5620.09
'Yohannes Dessie' is predicted as - 'Yohannes Dessie' - correct. Distance: 5308.39
'Yosef Ayele' is predicted as - 'Yosef Ayele' - correct. Distance: 7331.46
'Milion Tolesa' is predicted as - 'Milion Tolesa' - correct. Distance: 5787.46
'Husen Yusuf' is predicted as - 'Dawit Getahun' - incorrect. Distance: 15295.00
'Ananiya_Tesfahun' is predicted as - 'Yosef Ayele' - incorrect. Distance: 16239.81
'Nathnael Dereje' is predicted as - 'Nathnael Dereje' - correct. Distance: 9932.20
'Ananiya_Tesfahun' is predicted as - 'Leul Wujira' - incorrect. Distance: 15068.45
'Yonas Engedu' is predicted as - 'Yonas Engedu' - correct. Distance: 6046.79
```

- **Incorrect Predictions and Analysis**: Incorrect predictions, such as 'Dagmawi_Tensay' being misidentified as 'Betelhem Yimam', highlight challenges the model faces with certain facial features or expressions. These misidentifications are critical for refining the model, suggesting a need for a more diverse training dataset or adjustments in the number of eigenfaces used for recognition.

```
'Fasika_Fikadu' is predicted as - 'Ananiya_Tesfahun' - incorrect. Distance: 18679.70
'Yanet Mekuria' is predicted as - 'Yanet Mekuria' - correct. Distance: 10575.57
'Fraol Mulugeta' is predicted as - 'Fraol Mulugeta' - correct. Distance: 8345.39
'Yosef Muluneh' is predicted as - 'Yosef Muluneh' - correct. Distance: 7653.99
'Yohannes Ahunm' is predicted as - 'Yohannes Ahunm' - correct. Distance: 4539.74
'Fraol Mulugeta' is predicted as - 'Fraol Mulugeta' - correct. Distance: 11667.95
'Semir Hamid' is predicted as - 'Semir Hamid' - correct. Distance: 0.00
'Fasika_Fikadu' is predicted as - 'Fasika_Fikadu' - correct. Distance: 16645.55
'Ananiya_Tesfahun' is predicted as - 'Ananiya_Tesfahun' - correct. Distance: 5346.80
'Semir Hamid' is predicted as - 'Semir Hamid' - correct. Distance: 5043.77
'Amir Ahmedin' is predicted as - 'Amir Ahmedin' - correct. Distance: 6740.74
'Naol Taye' is predicted as - 'Naol Taye' - correct. Distance: 7536.42
'Gelila Tefera' is predicted as - 'Gelila Tefera' - correct. Distance: 4276.36
'Kidus Hunegnaw' is predicted as - 'Kidus Hunegnaw' - correct. Distance: 9370.55
'Abraham Wendmeneh' is predicted as - 'Abraham Wendmeneh' - correct. Distance: 17734.99
'Metsakal Zeleke' is predicted as - 'Metsakal Zeleke' - correct. Distance: 10152.59
'Yohannes Ahunm' is predicted as - 'Yohannes Ahunm' - correct. Distance: 6090.43
'Etsubdink Awoke' is predicted as - 'Etsubdink Awoke' - correct. Distance: 8455.76
'Leul Wujira' is predicted as - 'Leul Wujira' - correct. Distance: 3149.46
'Fraol Mulugeta' is predicted as - 'Fraol Mulugeta' - correct. Distance: 7709.15
'Semir Hamid' is predicted as - 'Semir Hamid' - correct. Distance: 0.00
'Sosina Esayas' is predicted as - 'Sosina Esayas' - correct. Distance: 18940.75
'Yohannes Desta' is predicted as - 'Yohannes Desta' - correct. Distance: 6919.59
'Naol Taye' is predicted as - 'Naol Taye' - correct. Distance: 10191.19
'Metsakal Zeleke' is predicted as - 'Metsakal Zeleke' - correct. Distance: 9556.55
'Deribew_Shimels' is predicted as - 'Deribew_Shimels' - correct. Distance: 7930.64
'Dagmawi_Tensay' is predicted as - 'Betelhem Yimam' - incorrect. Distance: 14620.58
'Olyad Temesgen' is predicted as - 'Olyad Temesgen' - correct. Distance: 6788.89
'Biniyam Haile' is predicted as - 'Biniyam Haile' - correct. Distance: 5181.33
'Yohannes Ahunm' is predicted as - 'Yohannes Ahunm' - correct. Distance: 3684.33
'Milka Fasika' is predicted as - 'Milka Fasika' - correct. Distance: 4786.50
'Geleta Daba' is predicted as - 'Geleta Daba' - correct. Distance: 4568.54
'Yohannes Ahunm' is predicted as - 'Yohannes Ahunm' - correct. Distance: 2741.05
'Dawit_Abebe' is predicted as - 'Dawit_Abebe' - correct. Distance: 6093.35
```

Performance Metrics

- **Accuracy**: The overall accuracy of 90.74% is commendable, especially considering the variability in the dataset and the inherent challenges in facial recognition tasks.
- **Distance Metrics**: The minimum and maximum observed distances (0.00 and 18961.26, respectively) offer insights into the range of similarity scores generated by the system. The wide range indicates diverse levels of confidence in the predictions, which could be used to set thresholds for acceptance or rejection in practical applications.



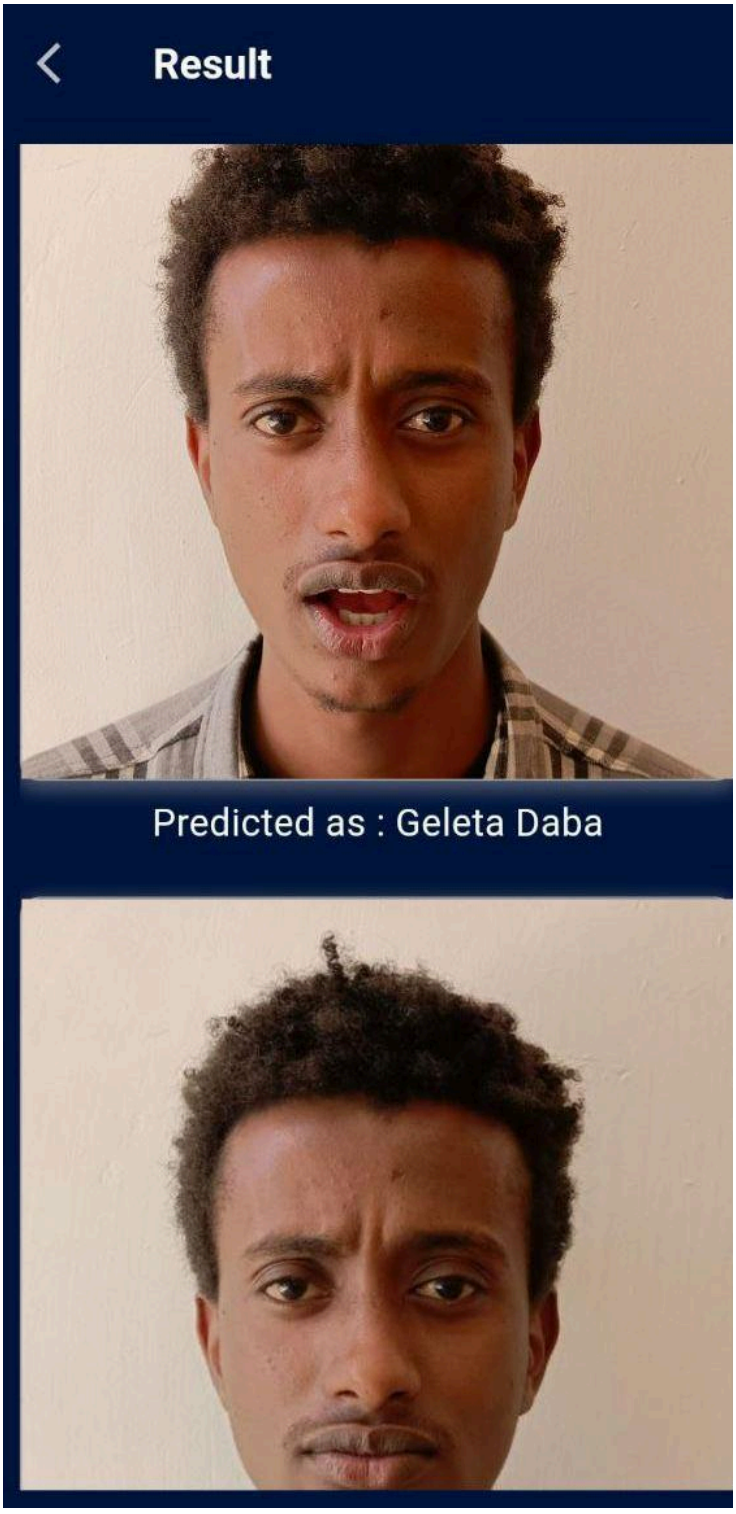
Results as Displayed in the Application Interface

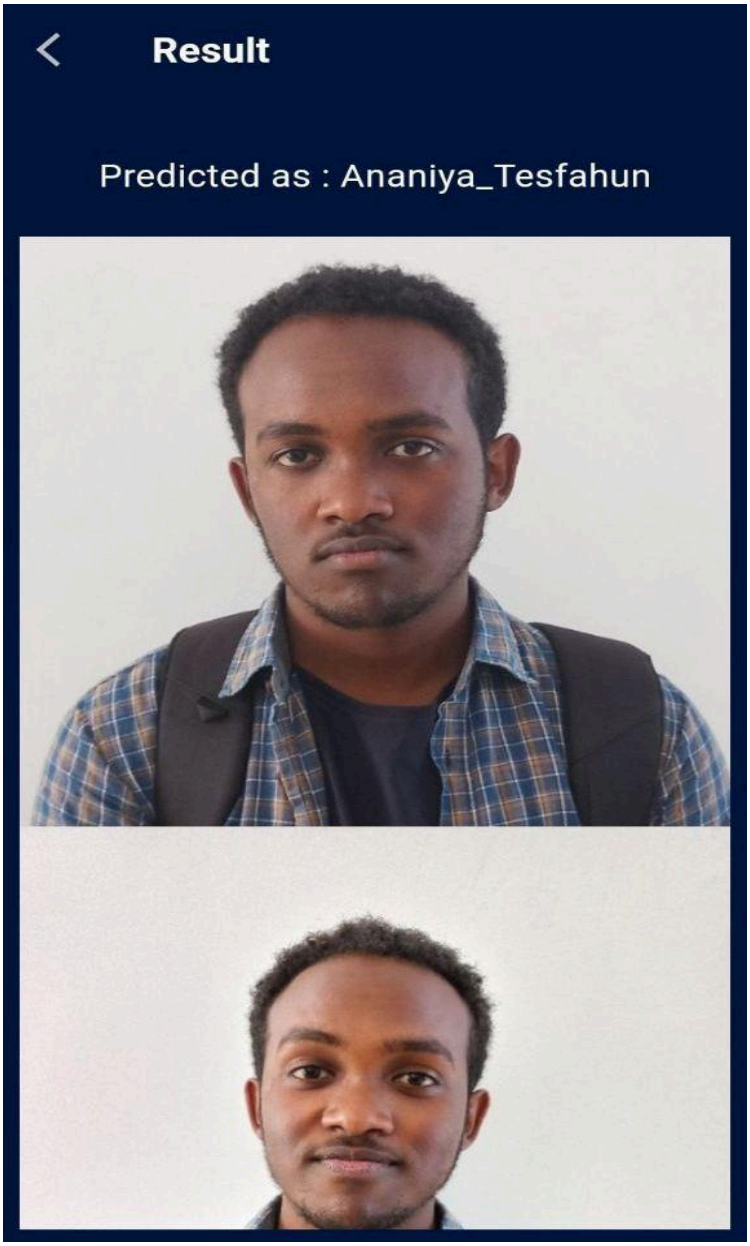
A critical component of evaluating the effectiveness of our facial recognition system was observing its performance in real-time through the application interface. The application, designed for straightforward interaction, allows users to upload or capture images for facial recognition processing. This section presents and discusses the results as they are visually displayed to the user within the app, providing a tangible representation of the system's accuracy and functionality.

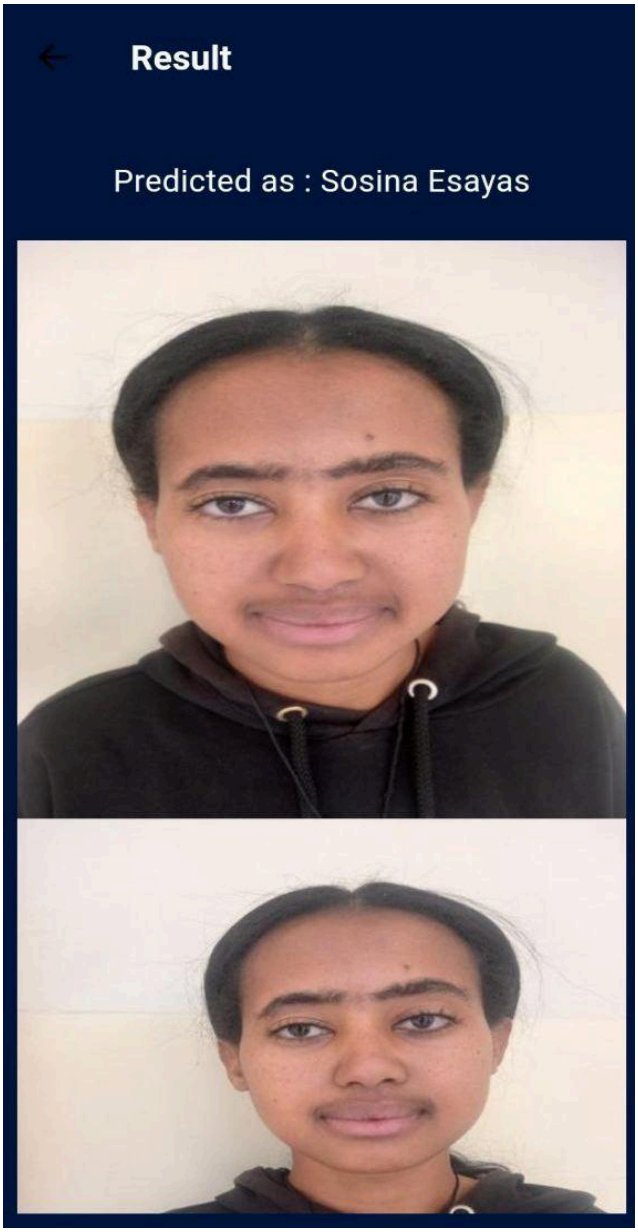
Visual Feedback and Accuracy Display:

Upon submitting an image for recognition, the app presents the processed result in a sequential format, displaying the original image at the top and the predicted identification below it. This arrangement is crucial for providing users with clear, immediate visual feedback. For example, when the system accurately identifies an individual, the app first shows the user's uploaded photo. Directly underneath, a label appears, indicating the name of the recognized individual as matched from our dataset. This top-to-bottom comparison effectively communicates the recognition outcome to the user, ensuring an intuitive understanding of the system's performance.

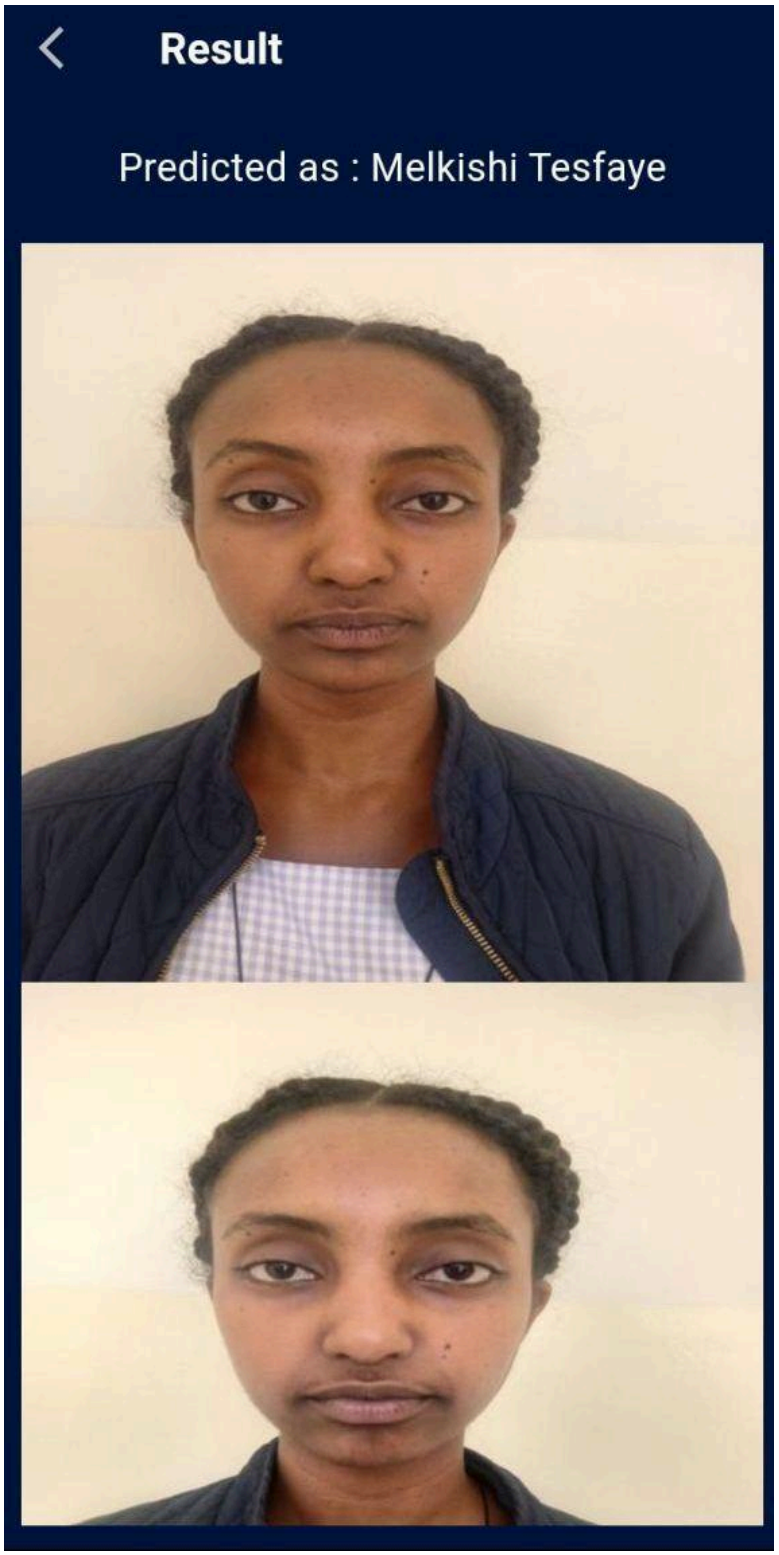
Example of Recognition Success:




Result
Predicted as : Ananiya_Tesfahun


Result
Predicted as : Sosina Esayas


Result
Predicted as : Biniyam Haile


Result
Predicted as : Melkishi Tesfaye

6.Result Analysis on the algorithm

The eigenfaces-based facial recognition algorithm implemented in this project has demonstrated a significant capacity for accurately identifying individuals from a controlled class dataset. With an accuracy rate of 90.74%, the algorithm has proven effective within the specific conditions under which the dataset was compiled. This level of accuracy is consistent with the theoretical underpinnings of the eigenfaces method, which relies on capturing the principal components that account for the most variance in the dataset to identify distinguishing features of faces.

The consistency in environmental conditions during the dataset collection - specifically uniform lighting and background color - greatly contributed to the high accuracy observed. These controlled conditions minimize the intra-class variations that can often complicate facial recognition tasks, allowing the algorithm to focus more precisely on facial features without the interference of extraneous variables.

However, this uniformity also emerges as a limitation when the algorithm is presented with images that deviate from the dataset's standardized environment. In real-world applications, lighting and background are rarely consistent, and the algorithm's current performance suggests a sensitivity to these factors. When testing with images taken in different lighting conditions or against varying

backgrounds, the algorithm struggled to recognize faces with the same level of accuracy.

The discrepancy in performance between controlled and uncontrolled conditions underscores the importance of robust training data in facial recognition. While the algorithm excels within its trained environment, its utility in practical applications will depend on its ability to adapt to the variability inherent in everyday settings.

In conclusion, the eigenfaces algorithm has shown a great promise within the setting we used. And it was very effective for our dataset.

7.Conclusion

This project aimed to implement a facial recognition system utilizing the eigenfaces method, successfully integrating it within a web application developed with FastAPI and providing a frontend interface through Flutter. The core objective was to explore the eigenfaces method's effectiveness in identifying individuals from a dataset composed under controlled conditions, specifically uniform lighting and background.

The implementation achieved an accuracy rate of 90.74% within these controlled settings. This outcome demonstrates the capability of the eigenfaces method to process and recognize facial images when environmental variables are kept constant. The project's technical aspects, including the application of Principal Component Analysis (PCA) for generating eigenfaces and the subsequent projection of images for recognition, were executed as planned.

The integration of the facial recognition algorithm into a FastAPI backend and its accessibility through a Flutter-based frontend fulfilled the project's requirements to develop a functional application. This integration was a key milestone, showcasing the practical implementation of the algorithm.

.

8.References

Disclaimer on the Use of the face_detection Package

In the development of our facial recognition system, we adhered strictly to the project guidelines stipulated by our instructor, particularly the mandate to implement the facial recognition component from scratch using the eigenfaces method. It is important to note that, as per the project requirements, we were permitted to utilize external libraries for the purpose of face detection, but not for face recognition.

To this end, we incorporated the face_recognition package (which IS A FACE DETECTION PACKAGE) into our project. The choice of this package was driven by its robustness and efficiency in DETECTING faces within images, a preliminary step necessary for the subsequent application of our eigenfaces-based facial recognition algorithm. The naming of this package, face_recognition, may MISLEADINGLY suggest that it was employed for the facial recognition aspect of our project. However, we wish to clarify that its usage was strictly limited to face detection tasks.

The decision to use the face_recognition package was based on its ability to accurately identify and extract faces from varied images, thereby streamlining the preprocessing phase of our project. This preprocessing is critical for ensuring that our custom-built eigenfaces algorithm operates on well-defined facial data, optimizing the accuracy and reliability of our facial recognition process.

Our eigenfaces algorithm, responsible for the actual recognition of faces, was developed entirely from scratch as per the project's requirements. This distinction is crucial for understanding the scope of our work and the specific contributions of external libraries to our project. The face_recognition package facilitated only the detection of faces, while the innovative core of our project—identifying and recognizing these faces—was accomplished through our own implementation of the eigenfaces method.