Mathematics and Applications of Facial Recognition using Eigenfaces

J. Abeje, K. Davies, A. Freeman, C. Henry, F. Rashid, B. K. Too
[1] Undergrad Research in Mathematics,
Howard University, Washington, D.C.

May 8th 2024

1 Abstract

This paper presents a facial recognition model that aims to identify parts of the face using a dataset of grayscale mugshot images representing a demographically diverse cohort. Utilizing Principal Component Analysis (PCA), this research describes eigenfaces by determining the eigenvectors from the "face space" of the facial data, which represent the main components of variance. These eigenfaces capture the facial variation itself rather than specific features like eyes or noses, which is crucial for reducing dimensionality and enhancing computational efficiency. The NIST mugshot dataset facilitates a comparative analysis to differentiate and classify images based on their racial features in an unsupervised manner. The model was evaluated on its ability to generalize across varied datasets, with particular attention to biases introduced during data preparation and model construction. Results indicate that while the model achieves a balance between precision and recall with an overall accuracy of 93.37%, it also exhibits potential biases in racial classification. This raises significant ethical, legal, and social concerns, particularly regarding racial profiling. The paper concludes with recommendations for improving model fairness through diversified data and advanced bias-mitigation techniques and stresses the importance of responsible deployment in real-world applications.

2 Introduction

In recent years, the utilization of facial recognition technology has expanded significantly, becoming integral to various applications across fields such as biometric security, digital identity verification, and domestic law enforcement[1]. However, the application of Machine Learning (ML) in facial recognition has foregrounded a range of serious ethical issues, especially those emanating from the potential biases the technology may entrench. This paper explores the application of Machine Learning techniques to the National Institute of Standards and Technology (NIST) Mugshot dataset[2] with the objective of predicting racial background, specifically classifying convicts as Black or non-Black. With these results, our model will show if there is a higher percentage of Black convicts or convicts.

In addition to PCA, Singular Value Decomposition (SVD) is a fundamental linear algebra technique used to enhance the eigenface method. This approach supports PCA's dimensionality reduction and enhances classification model robustness by handling data sparsity and noise effectively[3]. The eigenface method, refined by SVD and PCA, refines racial classification in law enforcement, aiming for higher accuracy and reduced bias in identifying individuals as Black or non-Black. By integrating SVD, the model better delineates facial characteristics, crucial for ethically applying machine learning in sensitive scenarios like racial classification[4].

With clean data and a model constructed on Jupiter Notebook, observing the accuracy of this model will allow us to determine the reliability of such techniques in place of human decision-making. The study aims to shed light on potential biases embedded within facial recognition algorithms and to assess the implications of using such models in real-world applications. By developing this model, we aim to provide a tool that enhances the precision of demographic analyses and supports the effective implementation of data-driven strategies in diverse research and applied settings.

3 Social Relevance

3.1 Evolution of Facial Recognition

Reflecting on the history of facial recognition, it is clear that this technology has evolved into a ubiquitous tool. In America, its early development is attributed to pioneers like Woodrow Bledsoe, Helen Chan Wolf, and Charles Bisson. Their research sought to utilize mathematical concepts to program computers for facial identification [5]. Although much of their work remains unpublished due to involvement with undisclosed government programs, their findings were foundational. Researchers Goldstein, Harmon, and Lesk built upon their work to identify 21 distinct markers of facial features, such as hair color and lip thickness, laying the groundwork for modern facial recognition.

A significant breakthrough came in 1991 with the discovery of eigenfaces by Turk and Pentland. Their methodology, involving principal component analysis to identify essential features, remains a cornerstone for many current facial recognition projects, including this investigation. The rapid development of facial recognition technology gained momentum after the 9/11 attacks, prompting increased interest and investment in this field. By the 2010s, facial recognition became commercially available through its use in social media platforms and its adoption for phone security in 2017 with the iPhone X.

Today, facial recognition technology is pervasive, influencing daily life whether through intentional interaction, like unlocking a phone, or unintentional, like being detected by public surveillance cameras. The evolution of facial recognition reflects not only technological advances but also the societal implications of its widespread adoption.

3.2 Racial Discrimination Facilitated by FRTs

Facial recognition is often associated with security and law enforcement, which is logical given its governmental origins. However, its usage has a troubling history of misuse against communities of color, particularly African Americans. Before the advent of facial recognition technology (FRT), the U.S. government employed various tactics to increase surveillance over civil rights organizations. The CounterIntelligence Program, COINTELPRO, used methods such as wiretapping, infiltration, legal harassment, and blackmail to disrupt the work of these organizations [6]. Notable Black activists, including Fred Hampton, Malcolm X, and Martin Luther King Jr., were targeted for increased surveillance. This historical misuse serves as a cautionary tale of what could happen if facial recognition technology remains unchecked by law, posing significant risks to the privacy of communities of color.

Recent instances have demonstrated continued abuse of FRT against Black Americans. As of April 2024, at least seven wrongful arrests of Black Americans have occurred due to faulty facial recognition results [7]. In 2020, at the height of the Black Lives Matter protests, local and federal law enforcement agencies deployed aerial surveillance technologies that included drones operating advanced facial recognition systems, like those created by Clearview.AI [8][9]. These systems invaded the privacy of individuals who peacefully protested, as their digital footprints were scrutinized to vilify them.

3.3 Connection to Current Study

The purpose of highlighting these racial concerns is not to condemn the development of facial recognition systems outright. Instead, it is to show how societal stigmas are amplified within these systems, creating a digital means of oppressing Black Americans. Our project aims to promote cultural sensitivity in the creation of these algorithms. It emphasizes the need for meticulous data cleaning and thoughtful algorithm design to ensure fair representation across diverse populations.

4 Background

4.1 Principal Component Analysis

Principal Component Analysis is widely used for face recognition and image processing by reducing the dimensionality of large datasets while preserving important information. In the Eigenface method for facial recognition, Principal Components Analysis (PCA) is employed to reduce the dimensionality of face image data, enhancing the efficiency and effectiveness of classification algorithms like Support Vector Machines (SVM). [10] in mean factor we will be using PCA because Centering is achieved by subtracting this mean face from each face image, which shifts the dataset such that the average face vector is zero. This step is crucial because PCA requires data to be centered around zero to accurately identify the principal components that capture the most significant variations in the data.[11]

$$\psi = \frac{1}{M}\sum_{i=1}^{M}\gamma \tag{1}$$

After calculating eigenvectors and eigenvalues, sort them by the magnitude of the eigenvalues. The top k eigenvectors are selected to form a new matrix W, which has the dimensions d × k. For a reduction to two dimensions, we select the top two eigenvectors. The final step involves using the matrix W to transform the original dataset into a new subspace. This transformation is achieved by projecting the original data onto the space defined by the selected eigenvectors:

$$Y = W^T X \tag{2}$$

This transformation results in a new dataset where the most significant variances are captured in the first few dimensions[3]. PCA helps in reducing the dimensionality by identifying the directions along which the variance of the data is maximized. It's particularly useful in pre-processing steps for machine learning algorithms that perform poorly with high-dimensional data due to the curse of dimensionality.

4.2 Eigenfaces

For creating a face recognition model with the eigenfaces method we are going to use PCA(Principal Component Analysis) and SVM(Support Vector Machine). Here are using SVM for the classification feature between ”Black” and ”NON-Black”.Principal Component Analysis (PCA) is used for dimensionality reduction of the image data before it is used for classification with a Support Vector Machine (SVM). Also, it's used to work with the bigger and more important eigenvectors for having more accuracy. For working with these models and methods we need to know about the mathematics working behind them and how this are recognizing the faces according to our required features.[11]

4.2.1 Transforming the Matrix into vectors

At first, the images we use to train the model are required to turn into a 1D vector from the 2D matrix. Each image, which is typically a two-dimensional grid of pixel values, is transformed into a one-dimensional vector by arranging the pixel values into a single, long sequence. This conversion facilitates mathematical operations, particularly those involving matrices, which are essential in the eigenfaces method. For example;

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ f & h \end{bmatrix} \tag{3}$$

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \tag{4}$$

$$\begin{bmatrix} e \\ f \\ g \\ h \end{bmatrix} \tag{5}$$

4.2.2 Mean Face Vector ψ

This is the mean face vector which we going to denote as ψ. It's the average of the training set image which we going to use later for getting a more accurate and effective calculation.[10] Here the mean face in the context of eigenfaces represents the average facial features of all the faces in the dataset. It's calculated by averaging each pixel's intensity across all the face images considered. This average image serves as a generic or central face that reflects the common features shared among the faces in the set, without emphasizing any individual's distinctive character. In the eigenfaces method, the mean face is used as a baseline or reference point. All individual face images are compared to this mean face to identify unique variations from the average, which helps in emphasizing the distinguishing features necessary for face recognition. Essentially, the mean face embodies the 'typical' facial features of the group under analysis.[12]

$$\psi = \frac{1}{M} \sum_{i=1}^{M} \gamma \tag{6}$$

Here γ is the one-dimensional vector that we made from our image matrix and M is the total number of images.

4.2.3 Difference between each image and average ϕ

The difference between each image and average ϕ is a critical step for facial recognition. This process, known as mean centering, involves subtracting the average face from the individual face image. Each image is then transformed by subtracting this average face, emphasizing the distinctive attributes crucial for face recognition. This mean-centered data is critical for the PCA as it ensures that PCA identifies the principal components (Eigenfaces) based on the most significant variances in face features, rather than mere averages. This process not only enhances the accuracy of the facial recognition system but also optimizes computational efficiency.[12]

$$\phi = \gamma - \psi \tag{7}$$

Here ψ is the mean average vector that we calculated in the previous step.

4.2.4 Covariance Matrix C

A covariance matrix expresses the covariance(Joint Variability) between pairs of variables in a dataset. For a dataset of N dimensions, the covariance matrix is an $N \times N$ matrix where each element (i,j) represents the covariance between the $i_t h$ and $j_t h$ dimensions of the dataset.[11]

In the context of images, if you consider each pixel value as a dimension, then the covariance matrix will tell you about the extent to which corresponding pixel values of different images vary from the mean concerning each other.

$$C = \frac{1}{M} AA^T \tag{8}$$

[12] Here A is the matrix that we built from the vector we get as the difference between the original image and the mean face vector, which is ϕ_1,ϕ_2 Now for simplicity, we calculate $A^T A$, matrix size ($M \times M$). Consider eigenvectors v_i of $A^T A$, so we can say ($A^T A$) v_i = $\mu_i v_i$ As it said before v_i is the eigenvector and now mu_i is the eigenvalue.[11] now we premultiply A on both sides, where we get $(A^T A)u_i$ = $\lambda_i u_i$. Here $Av_i = u_i$ and $\lambda_i = \mu_i$ It's important to keep in mind that the eigenvalue of AA^T and $A^T A$ are equal. It is because even though $A^T A$ and AA^T may look different and different matrices but they stem from the same underlying single value of A. That's why their eigenvalue will be equal. The covariance matrix in the eigenface method is crucial for analyzing how pixel intensities across different face images vary together, relative to the average (Mean Face). It represents the correlation between every pair of pixels across the set of images. [11]

4.2.5 Calculating Eigenvalues and Eigenvectors

After the covariance matrix, we come to calculate the eigenvalue and eigenvector which is the most important aspect in order to have our eigenface.

$$A - \lambda I \tag{9}$$

Here A is the covariance matrix that we gained from the previous step and lambda is known as the eigenvalue. I is the identity matrix. From here we get different eigenvalues which let us have different eigenvectors. For each eigenValue, we put the value in the equation of the matrix to get a different eigenvector, in one word the number of eigenvalues and eigenvectors are equal.[12]

4.2.6 Normalize eigenvectors

We normalize the vectors to ensure each vector has a unit length, which means they all have the same scale. This is crucial because it helps to compare the vectors fairly, focusing on their direction not on their magnitude. In the context of eigenfaces, normalization ensures that each eigenvector contributes equally to the facial representation, making the data based on the direction of the data's variability, not the scale.

$$\mathbf{u} = \frac{\mathbf{v}}{||\mathbf{v}||} \tag{10}$$

[12] So basically here V is the eigenvector that we calculated in the previous step and $||\mathbf{v}||$ is the absolute value of the eigenvector which we find out after finding the whole square of the sum of the square of the elements.

So this normalize vectors or $\mathbf{u_1}, \mathbf{u_2}, \mathbf{u_3}, \mathbf{u_4}$. So these are the eigenfaces of the image we have.[12] Now we are going to reconstruct the image with these eigenfaces, the mean faces of the training set, and with weight contribution of each eigenfaces.

In recognition, this normalizing ensures that when a new face is projected onto the space of eigenfaces, the similarity measurements and subsequent classification are not biased by the scale of the eigenfaces but rather by the actual content and structure of the face data. This helps accurately identify and recognize faces by their essential features.

4.2.7 Reconstruction

After normalizing the eigenvectors we came to the part of calculating the weight in order to reconstruct the image.

$$\gamma = \psi + [\mathbf{u_1}, \mathbf{u_2}, \mathbf{u_3}, \mathbf{u_4}] \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} \tag{11}$$

[12] Here ψ is the mean face vector as we mentioned earlier and $\mathbf{u}$ is the normalized vector as we calculated in the previous step. The most important thing is the weight W which we are going to calculate in this step. The reconstruction also looks like,

$$\gamma = \psi + \sum_{i_1}^{M} u_i w_i \tag{12}$$

For calculating the weight we have to work with,

$$w_k = u_k^T \phi \tag{13}$$

Here we select k's most important eigenvectors for k largest eigenvalues. [12]

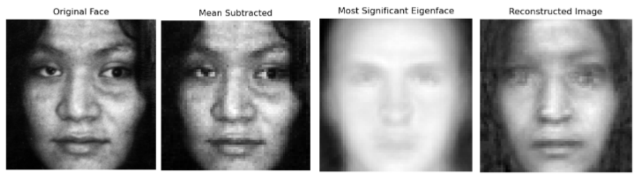


Figure 1: Visual of Eigenfaces Process

4.3 Linear Support Vector Machine (SVM)

In the context of Eigenface classification, Support Vector Machines (SVM) serve as a robust method for face recognition by efficiently handling high-dimensional input data. SVM operates by identifying a hyperplane that optimally separates different classes in a feature space, which, in the case of face recognition, would distinguish between different individuals. According to the paper by Theodoros Evgeniou and Massimiliano Pontil, SVM achieves this by transforming the original input space into a higher-dimensional space using a kernel function, thereby allowing for the handling of complex patterns in the data.[5] For face recognition, after applying PCA to reduce dimensionality and capture the essential features of faces (Eigenfaces), the SVM classifier then utilizes these features to classify new images. The SVM approach is particularly effective due to its ability to maximize the margin between different classes, thus enhancing its generalization ability and reducing the risk of overfitting, which is crucial for achieving high accuracy in facial recognition tasks.[11] Linear decision function:

$$f(x) = \mathbf{w} \cdot \mathbf{x} + b \tag{14}$$

W is the weight vector perpendicular to the hyperplane. X is the feature vector and b is the bias term.

Non-linear decision Function with kernel:

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b \tag{15}$$

Here the α_i are the lagrange multipliers obtained from solving the SVM optimization problem. These multipliers are non-zero only for the support vectors. y_i are the class labels of the training samples. x_i are the support vectors themselves. b is the bias term and $K(x_i, x)$ is the kernel function evaluated between the new data point of x and each support vector x_i

4.3.1 Classification Report

5 Methodology

5.1 Data Preproccesing

The dataset is split into training and testing sets to ensure the SVM model's reliability and generalizability. We put all Non-Black and Black individuals into folders, *"NONBLK"* and *"BLK"* respectively. We used these predetermined cases of individuals from this dataset being Black and Non-Black to train our model. We categorized if a person is Black or Non-Black based on three key Afro-eccentric features:

- **Skin Tone:** Evaluated the skin tone visible in the mugshot images, focusing on shades that typically correlate with Black or Non-Black ethnicities based on a predetermined color scale.
- **Hair Texture:** Identified typical hair textures associated with Black individuals, such as curly, coily, or kinky hair, versus straight or wavy textures more commonly found in Non-Black individuals.
- **Facial Structure:** Considered common facial structure traits such as nose width, lip fullness, and cheekbone prominence that are often cited in anthropological research as distinguishing features among different ethnic groups.

We also had to cut down on the amount of "NONBLK" front-facing images we included in our training model. We adjusted the dataset by reducing the number of "NONBLK" individuals to match the number of "BLK" individuals, resulting in 473 images for each group. images so that we could have an equal amount of images to represent both cases. Balancing the number of Non-Black and Black convict images in our training dataset was crucial to avoid introducing a class imbalance, which could skew the model's performance by making it biased towards the majority class.

Figure 2: Sample of Individuals from 'BLK' Folder

This ensures that our machine learning model learns to recognize patterns and make predictions with equal accuracy for both Black and Non-Black individuals,

Figure 3: Sample of Individuals from 'NON-BLK' Folder

thus improving its generalizability and fairness. However, this process introduces bias, as my group and I determining if a person is Black or Not leaves room for human error. The reliance on visual assessments and cultural interpretations can introduce personal biases and assumptions that might not accurately reflect an individual's ethnic background.

5.2 Testing Set

We put all single offenses and all multiple offenses convict images into folders *"single_and_multiple"* which will be used to test our model. There is no room for bias in this step as we already know if a convict has a single or multiple offense. This will allow us to observe our model's ability to accurately predict whether Black individuals have a higher population in single or multiple-offense convicts. These manual steps allow us to manipulate these images using code without having to sift through the various files they were placed in before.

Cleaning the dataset was an indispensable step. Without this, the PCA might extract features from metadata or noise -such as variations in image size and quality- rather than from the facial characters of interest. Similarly, the SVM classifier relies on the distinction of features to determine the separating hyperplane between classes accurately. Therefore, the preprocessing of images directly influences the efficacy of the eigenfaces and SVM in the subsequent stages of facial recognition.

5.2.1 Data Preparation:

Prior to integration into the machine learning workflow, the images undergo several preprocessing steps:

- **Image Standardization:** Given the variable sizes of the images, they are standardized to a uniform size to facilitate consistent processing across all data points.
- **Quality Enhancement:** Any necessary enhancements, such as contrast adjustment and noise reduction, are applied to improve image quality, which is vital for the robustness of the feature extraction phase.
- **Data Augmentation:** To address imbalances in the dataset, particularly concerning the underrepresentation of female subjects and the limited number of profile views, data augmentation techniques such as mirroring and slight rotation are employed to artificially expand the dataset.

5.2.2 Python Code: Image Pre-Proccessing

This section provides the Python code used for image preprocessing and processing images from a folder.

Load and Preprocess Image

```python
def load_and_preprocess_image(image_path):
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    if img is None:
        raise ValueError(f"Failed to load image from {image_path}.")

    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
    faces = face_cascade.detectMultiScale(img, scaleFactor=1.1, minNeighbors=5)
    if len(faces) == 0:
        return None, None

    x, y, w, h = faces[0]
    face_img = img[y:y+h, x:x+w]
    resized_img = cv2.resize(face_img, (64, 64))
    norm_img = (resized_img - np.mean(resized_img)) / np.std(resized_img)

    return norm_img.flatten(), face_img  # Return both the flattened normalized image and the original face image
```

Function Explanation

The function `load_and_preprocess_image` begins by attempting to load an image from a specified file path in grayscale format. This is done using OpenCV, a library for handling image processing tasks. If the image does not load, perhaps due to the file not existing or being unreadable, the function will halt and report an error indicating that the image could not be loaded.

Once the image is successfully loaded, the function uses a pre-trained model called a "haarcascade" to detect faces within the image. This model is designed to identify and locate faces based on the patterns of typical facial features. If no faces are detected, the function returns 'None', signifying that no further processing can be performed.

If a face is detected, the function proceeds by cropping the image to just the area containing the first face detected. It adjusts the size of this cropped portion to a standard 64x64 pixels, which is commonly used in image processing to ensure uniformity in input size for further analysis or machine learning models.

Following the resizing, the function normalizes the pixel values of the cropped image. Normalization adjusts the image data to have a mean value of zero and a standard deviation of one. This process is crucial as it helps in reducing discrepancies in image brightness and contrast, which might affect the performance of subsequent image processing tasks.

Finally, the function flattens the normalized image data into a one-dimensional array, which simplifies the use of this data in many machine learning algorithms by converting it into a format that can be easily fed into models. Additionally, the original cropped face image is also returned, allowing it to be used elsewhere if needed.

Process Images from Folder

```python
def process_images_from_folder(folder_path):
    images = []
    labels = []
    label_dict = {'BLK Single': 1, 'NONBLK Single': 0}

    for label in label_dict:
        class_folder = os.path.join(folder_path, label)
        print(folder_path)
        for filename in os.listdir(class_folder):
            if filename.lower().endswith(('.png', '.jpg', '.jpeg')):
                image_path = os.path.join(class_folder, filename)
                processed_img = load_and_preprocess_image(image_path)
                if processed_img[0] is not None:  # Ensure the processed image is valid
                    images.append(processed_img[0])  # Only append the flattened image
                    labels.append(label_dict[label])

    return np.array(images), np.array(labels)
```

Function Explanation

The function `process_images_from_folder` is designed to process multiple images stored in a directory for further use in data analysis or machine learning models. It takes a single argument, `folder_path`, which specifies the path to the directory containing the images.

Initially, two lists named `images` and `labels` are created to store processed image data and corresponding labels, respectively. A dictionary `label_dict` is also defined, which maps the category names of images to numeric labels — this aids in classification.

The function then iterates over each category defined in `label_dict`. For each category, it locates a subfolder within the main directory that matches the category's name. It lists all files in this subfolder and processes each image file that ends with extensions '.png', '.jpg', or '.jpeg'.

Using the function `load_and_preprocess_image`, each image is processed. If the image is valid (successfully processed), the flattened image data is appended to the `images` list and the corresponding label (from `label_dict`) to the `labels` list.

After processing all suitable images, the lists are converted to NumPy arrays to facilitate numerical operations and returned. This structured approach ensures that the images are not only processed for uniformity but also labeled correctly for any subsequent analytical tasks.

5.3 Math Methodology

5.3.1 Applying PCA

PCA is effectively used to reduce the dimensions of the data by transforming the original variable into a new set of variables(the principal components) which are uncorrelated and ordered so that the first few retain most of the variation present in all of the original variables.

How PCA works is first by standardization. The data is standardized to have a mean of zero and a standard deviation of one. Next compute the covariance matrix to understand how variables relate. Next, decompose the covariance matrix to its eigenvalues and eigenvectors. Then, select a subset of the eigenvectors as principle components; these are chosen based on their corresponding eigenvalues. Finally, project the original data onto the space spanned by the selected principal components.

5.3.2 Finding Eigenfaces

In this next section, we're going to look at how we calculate the eigenfaces of our images, which help us tell faces apart in our project. Think of eigenfaces as the main features that make each face unique, how you might recognize someone by their eyes or smile. We'll exemplify the mathematics with two small 2×2 matrices. This is equivalent to looking at the very basic building blocks of a much bigger picture. Along the way, we'll showcase eigenfaces we've found from the NIST mugshots. These pictures are key to how we teach our machine-learning model to see the difference between one person's face and another's. First we obtain face images $I(x, y)$ as training faces:

$$I(x,y) = \left\{ \begin{bmatrix} 1 & 1 \\ -2 & -3 \end{bmatrix}, \begin{bmatrix} 1 & -1 \\ -3 & 2 \end{bmatrix} \right\}$$

Represent every image I_i as a vector Γ_i. The image needs to be flattened to vector form because we need it to calculate the covariance later.

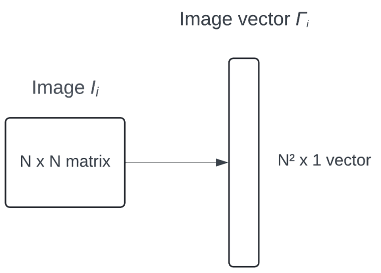


Figure 4: Converting an Image to a vector representation

Using the methodology in Figure 4, we will convert our 2×2 image matrix I_i into a 4×1 image vector Γ_i:

$$\Gamma_1 = \begin{bmatrix} 1 \\ -2 \\ 1 \\ -3 \end{bmatrix}_{4\times1}, \quad \Gamma_2 = \begin{bmatrix} 1 \\ 3 \\ -1 \\ 2 \end{bmatrix}_{4\times1}$$

Each face image is represented by the vector Γ_i. Here, Γ_1 and Γ_2 are the two example face image vectors we will utilize in the context of this paper.

Now that we have obtained the face image vectors, we can use them to calculate the average face. The computation of the average face is crucial in a facial recognition system because it provides a baseline against which to highlight individual differences through the subtraction of common features across the dataset. This subtraction process is effective at isolating those unique features of the face by canceling out variations due to generic, light, and orientation factors, and thereafter relatively simple in the extraction of the principal components (eigenfaces) that are vital for differentiating against faces. This step in the methodology supports the algorithms in developing dimensionality reduction, intending to enhance facial recognition system performance, focusing on significant variations peculiar to each face. The average face vector may calculated using the formula:

$$\Psi = \frac{1}{M} \sum_{i=1}^{M} \Gamma_i$$

We add the respective column elements and divide by the total number of face images to calculate the mean.

$$\Psi = \frac{\Gamma_1 + \Gamma_2 + \Gamma_3 + \ldots + \Gamma_M}{M}$$

Given that $M = 2$:

$$\Psi = \frac{1}{2} \begin{bmatrix} 1+1 \\ 2+3 \\ 1-1 \\ -3+2 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 2 \\ 1 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{1}{2} \\ 0 \\ -\frac{1}{2} \end{bmatrix}$$

We have now obtained the average face image Ψ.

Using the above methodology, we will now compute the average face from a dataset comprising 946 facial images. This average face synthesizes the common visual attributes present across the collective dataset, serving as a foundational reference in our analysis. By establishing this baseline, we effectively standardize the dataset, enabling the precise isolation of distinct facial features by comparing individual faces against this average composite. This normalization is instrumental in reducing variations attributable to generic attributes such as lighting and facial orientation, thereby enhancing the accuracy of subsequent facial feature extraction. The visualization of this average face not only provides a clear depiction of typical facial traits but also underpins the initial stage of our eigenface-based facial recognition framework.

Now we subtract average face image from the original image vectors. As explained above, this step brings out the differential features in each face that are vital for the later computation and process of recognizing the face.

$$\Phi_i = \Gamma_i - \Psi$$

Where Φ_i is the difference between each image and the average face image.

For our two face image vectors, we calculate Φ_1 and Φ_2 as follows:

$$\Phi_1 = \begin{bmatrix} 1 \\ -2 \\ 1 \\ -3 \end{bmatrix} - \begin{bmatrix} 1 \\ \frac{1}{2} \\ 0 \\ -\frac{1}{2} \end{bmatrix} = \begin{bmatrix} 0 \\ -\frac{5}{2} \\ 1 \\ -\frac{5}{2} \end{bmatrix}$$

$$\Phi_2 = \begin{bmatrix} 1 \\ 3 \\ -1 \\ 2 \end{bmatrix} - \begin{bmatrix} 1 \\ \frac{1}{2} \\ 0 \\ -\frac{1}{2} \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{5}{2} \\ -1 \\ \frac{5}{2} \end{bmatrix}$$

Each step differs from the average by $\Phi_i = \Gamma_i - \Psi$.

In the described methodology, the average face is computed from a dataset of training images by averaging pixel values across all images, thus forming a baseline representation of common facial traits. This baseline is utilized to accentuate individual differences by subtracting it from each face image in the test set. The resultant difference image highlights unique facial features critical for the facial recognition process by isolating deviations from the normative facial characteristics. This operation is visualized in Figure **??**, which presents the difference image obtained by subtracting the average face from the test image. The difference image, elucidates the distinct facial features by eliminating commonalities, thereby enhancing the facial recognition system's ability to focus on and analyze these unique attributes essential for accurate identification.

The covariance matrix plays a critical role in the eigenface approach by measuring the extent to which different pixel values in the set of facial images vary together, thereby identifying correlations across the dataset. This matrix forms the basis for identifying the eigenvectors that represent the most significant directions of data variability,or the "principal components" of the face data. These components are essenial for effectively capturing the distinguishing features of faces, facilitating enhanced facial recognition performance.

The covariance matrix is constructed as $C = \frac{1}{M}AA^\top$ because:

$$C = \frac{1}{M}\sum_{n=1}^{M}\Phi_n\Phi_n^\top = AA^\top$$

Where the matrix $A = [\Phi_1\Phi_2\ldots\Phi_M]$ of size $N^2 \times N^2$.

For our scenario where $M = 2$ and $N = 2$:

$$A = \begin{bmatrix} 0 & 0 \\ -\frac{5}{2} & \frac{5}{2} \\ 1 & -1 \\ -\frac{5}{2} & \frac{5}{2} \end{bmatrix}_{N^2\times M}, \quad A^\top = \begin{bmatrix} 0 & -\frac{5}{2} & 1 & -\frac{5}{2} \\ 0 & \frac{5}{2} & -1 & \frac{5}{2} \end{bmatrix}_{M\times N^2}$$

The size of the covariance matrix will be $AA^\top$ which is 4×4.

Calculating $AA^\top$

$$AA^T = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{25}{2} & -\frac{5}{2} & \frac{25}{2} \\ 0 & -\frac{5}{2} & \frac{1}{2} & -\frac{5}{2} \\ 0 & \frac{25}{2} & -\frac{5}{2} & \frac{25}{2} \end{bmatrix}$$

Therefore the covariance matrix will be

$$C = \frac{1}{2}AA^\top = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 6.25 & -2.5 & 6.25 \\ 0 & -2.5 & 1 & -2.5 \\ 0 & 6.25 & -2.5 & 6.25 \end{bmatrix}$$

Now, we need to find the eigenvalues and eigenvectors corresponding to the covariance matrix. Calculating Eigenvalues and eigenvectors is crucial for the eigenface method as they define the principal components that capture most of the significant variance in the dataset of face images. Eigenvalues help quantify the level of importance of each eigenvector in identifying how much of the data's variability is accounted for by any vector.This subsequently allows building a lower-dimensional feature space that retains essential characteristics required for effective face recognition and thus ensures the system focuses on the most informative aspects of the facial data.

To calculate the eigenvalues, we begin by forming a new matrix by subtracting λI from the diagonal entries of the given matrix:

$$\det(A - \lambda I) = \begin{vmatrix} -\lambda & 0 & 0 & 0 \\ 0 & 6.25-\lambda & -2.5 & 6.25 \\ 0 & -2.5 & 1-\lambda & -2.5 \\ 0 & 6.25 & -2.5 & 6.25-\lambda \end{vmatrix}$$

The characteristic polynomial derived from the matrix is given by:

$$13.5\lambda^3\left(\frac{2}{27}\lambda - 1\right)$$

Simplifying this, we factor out the terms to identify the roots:

$$\lambda(\lambda^3)\left(\frac{2}{27}\lambda - 1\right) = 0$$

From the polynomial, $\lambda = 0$ is a root with a multiplicity of three:

$$\lambda_1 = 0, \quad \lambda_2 = 0, \quad \lambda_3 = 0$$

Given the equation derived from the characteristic polynomial:

$$\frac{2}{27}\lambda = 1$$

Isolate λ by multiplying both sides of the equation by the reciprocal of $\frac{2}{27}$:

$$\lambda = \frac{27}{2}$$

Therefore, the eigenvalue λ that satisfies this equation is $\frac{27}{2}$. This is the non-zero eigenvalue, while the zero eigenvalue appears with a multiplicity of three, leading to the following eigenvalues for the matrix:

$$\begin{aligned}\lambda_1 &= 0,\\ \lambda_2 &= 0,\\ \lambda_3 &= 0,\\ \lambda_4 &= \frac{27}{2}.\end{aligned}$$

Now, we may calculate the eigenvectors.

The eigenvalues of the covariance matrix are $\lambda_1 = \lambda_2 = \lambda_3 = 0$ and $\lambda_4 = \frac{27}{2}$. We find the corresponding eigenvectors by substituting these values into the matrix $A - \lambda I$ and reducing to row echelon form to solve for the null space.

For $\lambda = 0$: The matrix $A - 0I = A$ is:

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 6.25 & -2.5 & 6.25 \\ 0 & -2.5 & 1 & -2.5 \\ 0 & 6.25 & -2.5 & 6.25 \end{bmatrix}$$

Applying Gaussian Elimination, (using MatrixCalc's eigenvector calculator [13])
The reduced row-echelon form is:

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & -0.4 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

From this, the eigenvectors for $\lambda = 0$ are:

$$v_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$v_2 = \begin{bmatrix} 0 \\ \frac{2}{5} \\ 1 \\ 0 \end{bmatrix}$$

$$v_3 = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 1 \end{bmatrix}$$

For the non-zero eigenvalue $\lambda = \frac{27}{2}$: Substitute $\lambda = \frac{27}{2}$ into $A - \lambda I$ to get:

$$\begin{bmatrix} -\frac{27}{2} & 0 & 0 & 0 \\ 0 & 6.25 - \frac{27}{2} & -2.5 & 6.25 \\ 0 & -2.5 & 1 - \frac{27}{2} & -2.5 \\ 0 & 6.25 & -2.5 & 6.25 - \frac{27}{2} \end{bmatrix}$$

Applying Gaussian elimination leads to:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -\frac{2}{5} & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Therefore the eigenvector for $\lambda = \frac{27}{2}$ is:

$$v_4 = \begin{bmatrix} 0 \\ 1 \\ -\frac{2}{5} \\ 1 \end{bmatrix}$$

To summarize:

$$\lambda_1 = 0, \quad v_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \text{(Eigenvector } v_1\text{)}$$

$$\lambda_2 = 0, \quad v_2 = \begin{bmatrix} 0 \\ \frac{2}{5} \\ 1 \\ 0 \end{bmatrix} \quad \text{(Eigenvector } v_2\text{)}$$

$$\lambda_3 = 0, \quad v_3 = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 1 \end{bmatrix} \quad \text{(Eigenvector } v_3\text{)}$$

$$\lambda_4 = \frac{27}{2}, \quad v_4 = \begin{bmatrix} 0 \\ 1 \\ -\frac{2}{5} \\ 1 \end{bmatrix} \quad \text{(Eigenvector } v_4\text{)}$$

The eigenvalues and eigenvectors calculated here provide vital insights into the distribution of variance within the dataset. Eigenvectors v_1, v_2, and v_3 are associated with eigenvalues of 0. This indicates that these vectors do not contribute to the variance of the dataset. Instead, they represent along where there is no significant variation.

In contrast, the eigenvector v_4 with the eigenvalue of $\frac{27}{2}$ stands out as the principal component as it captures the highest variance within the face images. This particular eigenvalue suggests that the eigenvector v_4 is the most crucial direction for differentiating faces in the dataset because it holds key features that differentiate one face from another. Therefore, for facial recognition or compression purposes, focusing on eigenvector v_4 would be most effective.

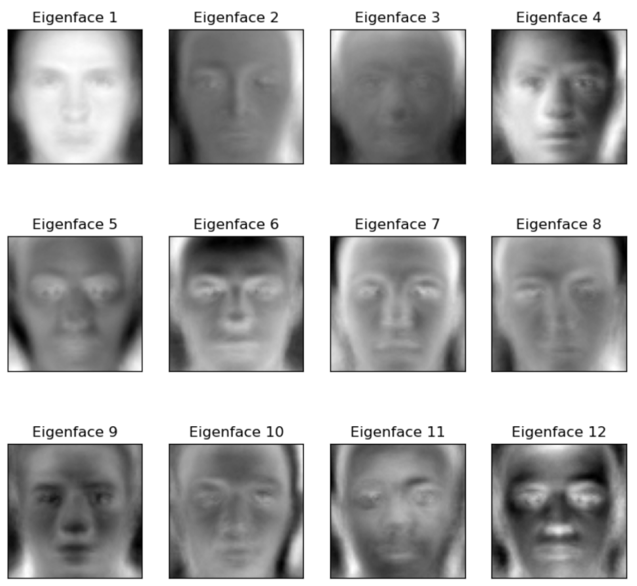


Figure 5: Our Model displaying the first 12 eigenfaces

The images are scaled to have values between 0 and 1, and the mean face is subtracted to center the data around zero. PCA is used to compute the eigenfaces, which are essentially the eigenvectors of the data's covariance matrix. The principal components (eigenfaces) are reshaped back into the original image dimensions and displayed. To interpret and visualize the results of PCA, the eigenfaces (principal components) are reshaped back into the 64×64 pixel format and displayed as images. Each eigenface can be seen as a ghostly representation of the facial features that characterize the most significant variations across the dataset. The visualization is performed using matplotlib, providing insights into what features the PCA model is capturing, which may include variations in facial attributes related to different racial characteristics.

Before proceeding further, we need to normalize our eigenvectors/eigenfaces. To normalize eigenvectors using the Euclidean distance (or to make them unit vectors), we may use the normalization formula where each component of the eigenvector is divided by the magnitude (or norm) of the eigenvector. The Euclidean norm, also known as the L^2 norm, of a vector v is calculated as the square root of the sum of the squares of its components.

For an eigenvector $v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$, its Euclidean norm $\|v\|$ is given by:

$$\|v\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}$$

To normalize the vector v, each component v_i of the vector is divided by $\|v\|$:

$$v_{\text{normalized}} = \frac{1}{\|v\|} v = \begin{bmatrix} \frac{v_1}{\|v\|} \\ \frac{v_2}{\|v\|} \\ \vdots \\ \frac{v_n}{\|v\|} \end{bmatrix}$$

This results in a vector of unit length, ensuring that its Euclidean norm is 1, which is a common requirement for many mathematical procedures, including those in PCA and machine learning algorithms where direction is important but magnitude must be standardized.

Normalize eigenvector v_1:

$$\|v_1\| = \sqrt{1^2 + 0^2 + 0^2 + 0^2} = 1$$

Thus, the normalized eigenvector u_1 is:

$$u_1 = \frac{1}{\|v_1\|} v_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Normalize eigenvector v_2:

$$\|v_2\| = \sqrt{0^2 + \left(\frac{2}{5}\right)^2 + 1^2 + 0^2} = \sqrt{\frac{29}{25}}$$

Thus, the normalized eigenvector u_2 is:

$$u_2 = \frac{1}{\sqrt{\frac{29}{25}}} \begin{bmatrix} 0 \\ \frac{2}{5} \\ 1 \\ 0 \end{bmatrix} = \frac{5}{\sqrt{29}} \begin{bmatrix} 0 \\ \frac{2}{5} \\ 1 \\ 0 \end{bmatrix}$$

Normalize eigenvector v_3:

$$\|v_3\| = \sqrt{0^2 + (-1)^2 + 0^2 + 1^2} = \sqrt{2}$$

Thus, the normalized eigenvector u_3 is:

$$u_3 = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 \\ -1 \\ 0 \\ 1 \end{bmatrix}$$

Normalize eigenvector v_4:

$$\|v_4\| = \sqrt{0^2 + 1^2 + \left(-\frac{2}{5}\right)^2 + 1^2} = \sqrt{\frac{54}{25}}$$

Thus, the normalized eigenvector u_4 is:

$$u_4 = \frac{1}{\sqrt{\frac{54}{25}}} \begin{bmatrix} 0 \\ 1 \\ -\frac{2}{5} \\ 1 \end{bmatrix} = \frac{5}{\sqrt{54}} \begin{bmatrix} 0 \\ 1 \\ -\frac{2}{5} \\ 1 \end{bmatrix}$$

The resulting eigenvectors u_1, u_2, u_3, and u_4 are the eigenfaces of the training dataset.

By normalizing the eigenvectors v_1, v_2, v_3, and v_4 to u_1, u_2, u_3, and u_4, every vector is normalized into unit eigenfaces. This is an essential step for effective Principal Component Analysis (PCA) in facial recognition. The normalization ensures that each eigenface has a unit length, establishing a consistent base for comparison and computation within the feature space. This uniformity is vital because it removes the effect of varying magnitudes among the original eigenvectors and ensures that every principal component has equitable contributions.

Normalization leads to several key benefits in the PCA framework:

- **Computational Simplicity**: Calculations involving dot products and other linear algebraic operations related to calculations with unit vectors become simplified, thus increasing computational efficiency.
- **Enhanced Interpretability**: The transformation into unit vectors assists in interpreting the results of the PCA by clearly indicating the relative importance of each principal component, making it easier to analyze the effect on the data's structure.
- **Uniform Contribution**: Normalization ensures that the contribution of each eigenface is based solely on its ability to collect the variance within the data, rather than on differences regarding the vector magnitude.

These unit vectors, or eigenfaces, represent the most significant patterns of variation within the dataset. For instance, specific eigenfaces may encapsulate essential features such as variations due to lighting conditions or distinct facial attributes like glasses or facial hair. This dimensional reduction achieved through PCA allows facial recognition systems to operate more efficiently by focusing on these key features, thus simplifying the complexity inherent in high-dimensional data[**?**].
In the context of our mugshots dataset, this whole mathematical process of calculating the eigenvalues, eigenvectors, and normalizing them, is simplified by utilizing scikit-learn's PCA library.

5.4 Image Weights

Calculating the weights in the context of eigenfaces and PCA is a fundamental step with several critical applications and benefits in facial recognition and other areas of image processing. The primary objective of calculating weights is to reduce the dimensionality while retaining pertinent features of the data. When high-dimensional face images are projected into a lower-dimensional space defined by eigenfaces, the resulting weights encapsulate the essential characteristics of each face. This reduction, therefore, makes data handling and analysis efficient. The weights for a face image Γ_i can be obtained by projecting the average-face image Φ_i onto each eigenface u_i:

$$w_i = u_i^T \Phi_i$$

where, $\Phi_i = \Gamma_i - \Psi$, is the mean-face image, and u_i^T is the transpose of the eigenface. Thus, the four weights can be calculated as follows:

$$w_1 = u_1^T \Phi_1, w_2 = u_2^T \Phi_1, w_3 = u_3^T \Phi_1, w_4 = u_4^T \Phi_1$$

Calculating w_1:

$$w_1 = u_1^T \Phi_1 = \frac{1}{\sqrt{1}} \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ -\frac{5}{2} \\ 1 \\ -\frac{5}{2} \end{bmatrix}$$

$$w_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ -\frac{5}{2} \\ 1 \\ -\frac{5}{2} \end{bmatrix}$$

$$w_1 = 0$$

Thus, the weight w_1 for the first eigenface u_1 with respect to the image vector Φ_1 is 0. Similarly, to calculate the weight w_2:

$$w_2 = u_2^T \Phi_1 = \frac{5}{\sqrt{29}} \begin{bmatrix} 0 & \frac{2}{5} & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ -\frac{5}{2} \\ 1 \\ -\frac{5}{2} \end{bmatrix} = 0$$

Calculating w_3:

$$w_3 = u_3^T \Phi_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ -\frac{5}{2} \\ 1 \\ -\frac{5}{2} \end{bmatrix} = 0$$

Calculating w_4:

$$w_4 = u_4^T \Phi_1 = \frac{5}{\sqrt{54}} \begin{bmatrix} 0 & 1 & -\frac{2}{5} & 1 \end{bmatrix} \begin{bmatrix} 0 \\ -\frac{5}{2} \\ 1 \\ -\frac{5}{2} \end{bmatrix} = -\frac{\sqrt{54}}{2}$$

Calculating the weights of the second image I_2

Calculating w_1:

$$w_1 = u_1^T \Phi_2 = \frac{1}{\sqrt{1}} \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ \frac{5}{2} \\ -1 \\ \frac{5}{2} \end{bmatrix} = 0$$

Calculating w_2:

$$w_2 = u_2^T \Phi_2 = \frac{5}{\sqrt{29}} \begin{bmatrix} 0 & \frac{2}{5} & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ \frac{5}{2} \\ -1 \\ \frac{5}{2} \end{bmatrix} = 0$$

Calculating w_3:

$$w_3 = u_3^T \Phi_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ \frac{5}{2} \\ -1 \\ \frac{5}{2} \end{bmatrix} = 0$$

Calculating w_4:

$$w_4 = u_4^T \Phi_2 = \frac{5}{\sqrt{54}} \begin{bmatrix} 0 & 1 & -\frac{2}{5} & 1 \end{bmatrix} \begin{bmatrix} 0 \\ \frac{5}{2} \\ -1 \\ \frac{5}{2} \end{bmatrix} = \frac{\sqrt{54}}{2}$$

Using PCA and the eigenface method, the weights w_1, w_2, w_3, and w_4 were calculated for two different facial images, providing significant insights into the facial features captured by the eigenfaces. For both face images, the weights w_1, w_2, and w_3 were zero, indicating that the first three eigenfaces do not capture any variance relative to the mean face. This suggests that these eigenfaces do not contribute to differentiating these images from the average face. In contrast, the fourth eigenface weight w_4 is non-zero, indicating a significant variance captured from the images relative to the average face.

The magnitude of w_4 for the first image is $-\frac{\sqrt{54}}{2}$ while for the second image, it is $\frac{\sqrt{54}}{2}$. These values indicate that the fourth eigenface captures distinct and contrasting facial characteristics that are not represented by the first three eigenfaces. The contrast in the sign and magnitude of w_4 between the two images underscores the eigenface's sensitivity to specific features, suggesting its potential utility in distinguishing between different individuals or expressions. This attribute is particularly useful in facial recognition and classification tasks, where identifying unique facial features is crucial. The presence of both negative and positive weights of equal magnitude highlights the eigenface's role in enhancing the discriminative power of the analysis, providing a robust measure of facial feature variance across different images.

In the analysis, Principal Component Analysis (PCA), facilitated through Scikit-Learn's PCA library, simplifies the spatial representation of facial images by projecting them onto a lower-dimensional space known as eigenface space. This methodology is vital, as it not only simplifies the dataset but also preserves significant facial features that define variability across the sample set. If we go back to our four eigenfaces, we can also calculate their weights, respectively, in a more straightforward manner through Scikit-Learn's PCA library. Specifically, the four pre-centered eigenfaces from the training set are transformed using this PCA model, yielding a set of weights for each image that quantitatively articulates how much each eigenface contributes to reconstructing the original image. These weights offer a numeric and visual method to evaluate the impact of specific facial features captured by the eigenfaces.

5.5 Employing a Support Vector Machine

In this project, the Support Vector Machine (SVM) is utilized as the primary classification tool, distinguishing between facial images categorized as either Black or non-Black. This binary classification employs the eigenface method for advanced feature extraction, where facial characteristics are reduced into a set of principal components—eigenfaces. These components are transformed into numerical weights, which the SVM uses to effectively differentiate between the two racial categories. The derivation of eigenfaces and their associated weights, as detailed previously, serves as the foundation for the SVM's input features. By using Principal Component Analysis (PCA), the high-dimensional data is condensed into a compact form, preserving essential information while reducing redundancy. This dimensional reduction facilitates a more efficient and accurate classification by the SVM.

SVM operates on the fundamental principle of structural risk minimization, aimed at finding a hyperplane that maximally separates classes in a feature space. The underlying premise of SVM involves constructing a hyperplane or set of hyperplanes in a high or infinite-dimensional space, which can be used for classification, regression, or other tasks.

General Formulation

Given a set of training vectors $x_i \in R^n$, $i = 1, \ldots, m$ with labels $y_i \in \{1, -1\}$, SVM solves the following primal optimization problem:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \xi_i$$

$$\text{subject to } y_i(\mathbf{w}\cdot\phi(x_i)+b)\geq 1-\xi_i, \quad \xi_i\geq 0$$

where:

- $\mathbf{w}$ is the normal vector to the hyperplane.
- b is the bias term, determining the offset of the hyperplane from the origin.
- ξ_i are slack variables that allow margin violations (soft margin).
- C is a regularization parameter that balances the margin maximization and loss.

The function $\phi(x_i)$ denotes a feature space transformation, and $\mathbf{w}\cdot\phi(x_i)$ represents the dot product in that space.

Kernel Trick

The kernel trick involves substituting the inner products in the feature space with a kernel function $K(x_i,x_j)=\phi(x_i)\cdot\phi(x_j)$. This allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space. The kernel function can be linear, polynomial, radial basis function (RBF), or sigmoid.

$$K(x_i,x_j)=\exp(-\gamma\|x_i-x_j\|^2)$$

for the RBF kernel, where γ is a parameter that defines the spread of the kernel.

Linear SVM Specialization

In the context of the project, after PCA, the data are expected to exhibit enhanced linear separability in the reduced feature space. Thus, a linear kernel is selected, simplifying the kernel function to:

$$K(x_i,x_j)=x_i\cdot x_j$$

The decision function for the linear SVM classifier is:

$$f(x)=\text{sgn}(\mathbf{w}^Tx+b)$$

This represents a linear decision surface where $\mathbf{w}$ is adjusted along with b to create the optimal separating hyperplane.

Implementation and Evaluation

Code Implementation

The SVM classifier is implemented using the scikit-learn library, as shown in the following Python code:

```python
from sklearn.svm import SVC
clf = SVC(kernel='linear')
clf.fit(X_train_pca, y_train)
y_pred = clf.predict(X_test_pca)
```

In our research, we integrated the SVM classifier with Principal Component Analysis (PCA) for preprocessing to manage the high dimensionality of facial images. This combination is crucial for reducing the computational complexity and enhancing the classifier's efficiency:

```python
# Initialize the SVM classifier with specific
    parameters
svm = SVC(kernel='linear', C=0.1, gamma
    =0.001, probability=True)
```

The SVM is configured with a linear kernel and parameters $C=0.1$ and $\gamma=0.001$, tailored to balance complexity and performance. The ‘probability=True‘ parameter allows the generation of probability estimates, which are calibrated using k-fold cross-validation:

```python
# Calibrate the SVM classifier
clf = CalibratedClassifierCV(svm, cv=5)
clf.fit(X_train_pca, y_train)
```

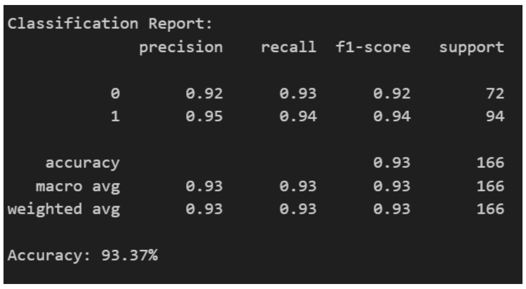
Calibration and Validation

To ensure the reliability of probability outputs from SVM, which does not naturally output calibrated probabilities, we employ a ‘CalibratedClassifierCV‘. This method enhances the interpretability of the model's predictions:

```python
# Evaluate the model
y_pred = clf.predict(X_test_pca)
print(classification_report(y_test, y_pred))
```

6 Results

The performance of the SVM model, trained using eigenfaces as features, is quantitatively evaluated in the classification report provided below. The report details precision, recall, F1-score for each class, and the overall model accuracy, macro average, and weighted average scores.



```
Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.93      0.92        72
           1       0.95      0.94      0.94        94

    accuracy                           0.93       166
   macro avg       0.93      0.93      0.93       166
weighted avg       0.93      0.93      0.93       166

Accuracy: 93.37%
```

Figure 6: Classification Report of our Model

6.1 Classification Report Analysis

- **Precision:** The model shows high precision, with scores of 0.92 for the 'Non-Black' class and 0.95 for the 'Black' class. This indicates that the model is correct 95% and 92% of the time when predicting 'Black' and 'Non-Black' faces, respectively. High precision is critical in applications where the cost of a false positive is significant.
- **Recall:** The recall scores are 0.93 for 'Non-Black' and 0.94 for 'Black'. These scores indicate the model's effectiveness in identifying all relevant instances of each class, crucial for systems where failing to detect a positive instance can have serious consequences.
- **F1-Score:** With f1-scores of 0.92 for 'Non-Black' and 0.94 for 'Black', the model demonstrates an excellent balance between precision and recall, crucial for maintaining accuracy in varied practical scenarios.
- **Support:** The support numbers show actual occurrences of each class in the test set—72 for 'Non-Black' and 94 for 'Black', indicating a slight class imbalance that the model handles effectively.
- **Overall Accuracy:** The overall model accuracy is 93.37%, demonstrating the model's effectiveness across both racial categories, making it a reliable tool in demographic analysis and other fields.

6.2 Cross-Validation Insights

Cross-validation results:

- Scores: 89.47%, 91.73%, 89.47%, 92.42%, and 93.18%
- Mean Cross-Validation Accuracy: 91.26%

Cross validation is a common way for researchers to test their machine learning models. Cross validation can provide insightful prediction as to how the model would preform with unseen data. This evaluation is done by separating the available data into subsets while reserving one to be a validation set and the others as a training set. This method is repeated yielding varying cross validation values.

Our mean cross validation accuracy is 91.26%. This high value suggests that the model would preform well with unseen data improving the reproducibility of it. In addition, the set of the cross validation values all exceed 89% which indicates consistency of performance. This allows us to be confident that the model can be taken and applied to generalized data sets outside of the one we collected.

7 Discussion

The model's reproducibility hinges on its ability to generalize across different datasets beyond the controlled environment of the NIST Mugshot dataset. Testing the model on a broader range of datasets with variations in image quality, lighting, and demographic distributions would provide a better understanding of its generalizability. Consistent application of preprocessing steps such as image standardization and eigenfaces extraction is crucial to maintain performance across varied datasets.

- **Limitations:** The main limitation of our model is its binary classification system, which reduces the broad spectrum of human racial diversity to simply 'Black' and 'Non-Black' categories. This oversimplification does not accurately reflect the complexity of race and ethnicity and could be expanded to include additional classifications. This is further complicated by manual adjustments in the dataset balance, potentially skewing the SVM's learning process. The inherent biases of the SVM algorithm need to be carefully managed through regularization techniques and hyperparameter tuning to avoid overfitting or underfitting.

  We also recognize that our data cleaning process may reflect our own personal biases. Moreover, the original dataset's predominance of white male individuals does not represent the demographic diversity of most jail populations or communities where facial recognition models are commonly used. This demographic imbalance reduces the dataset size and limits our ability to optimize the model fully.
- **Ethical Considerations:** Privacy and surveillance concerns are paramount with facial recognition technologies. The use of racial classification systems raises significant ethical questions, especially

if employed in contexts where they could reinforce racial profiling or contribute to discriminatory practices.

- **Legal Implications:** The deployment of facial recognition must comply with data protection laws, such as the GDPR or various state laws in the United States, which regulate the collection and use of biometric data. Transparency and accountability mechanisms are essential.

- **Social Impact:** The public acceptance of facial recognition technologies varies. Misidentifications, particularly of individuals from minority groups, could exacerbate social tensions and erode trust in institutions utilizing such technology.

- **Possibilities for Further Development** Incorporating a more diverse dataset could reduce bias and improve the model's generalizability. Exploring complex algorithms or deep learning architectures may also enhance accuracy and efficiency. Additionally, developing methods to make the model's decision-making process more transparent could help identify inherent biases and make the technology more accessible to a broader audience. As this technology evolves, we must also carefully consider its ethical implications. Clear guidelines and frameworks for the responsible use of racial classification technology are crucial.

  We encourage other researchers to replicate this methodology and use it as a foundation. Additional classifiers, covering other races and genders, could be added. We are interested to see if this model can accurately classify moving targets, not just still images. Researchers should engage with this process to cultivate cultural sensitivity when developing algorithms meant for human interaction. Taking the time to develop this cultural consciousness could alleviate many of the issues currently observed in facial recognition models.

While the eigenface and SVM-based model shows potential for classifying individuals based on racial characteristics with reasonable accuracy, careful management of its application is crucial, considering the broader ethical, legal, and social implications. Future work should aim to improve the model's fairness by incorporating a more diverse dataset and employing bias-minimization techniques. Engaging with stakeholders including policymakers, civil rights groups, and the public is essential to responsibly navigate the challenges posed by facial recognition technologies.

8 Concluding Statements

The results of this investigation demonstrate the importance of prioritizing racial considerations when developing machine learning algorithms for public use. Without these considerations, such algorithms risk perpetuating biases that disadvantage minority groups and favor the majority. Our model, which achieved an accuracy of 93.37%, shows how artificial intelligence can produce predictions that reflect the intentions of its creators. This serves as a reminder that models intended for specific communities should be developed by diverse teams to ensure fair and accurate representation.

Our eigenface model using SVM to classify individuals by race from grayscale images represents a significant step toward diversifying the digital landscape. Expanding this classifier range to include other racial categories and gender would further enhance inclusivity. Maintaining social awareness and cultural sensitivity is essential to refining classification models so that they meet public needs more effectively.

We urge researchers to remain socially conscious when creating algorithms that impact people's lives. By developing culturally conscious models, we can address current challenges in facial recognition systems and ensure that these tools serve all communities equitably.

References

[1] Kaspersky, “What is facial recognition? - definition,” https://usa.kaspersky.com/resource-center/definitions/what-is-facial-recognition, 2023, accessed: 2024-04-25.

[2] “Nist special database 18 – mugshot identification database,” https://www.nist.gov/srd/nist-special-database-18, 2014, accessed: 2023-04-17.

[3] O. Alter, P. O. Brown, and D. Botstein, “Singular value decomposition for genome-wide expression data processing and modeling,” *Proceedings of the National Academy of Sciences*, vol. 97, pp. 10 101–10 106, 2000.

[4] A. K. Jain, P. Flynn, and A. A. Ross, *Handbook of Biometrics*. New York, NY: Springer, 2005.

[5] E. Sullivan, “Facial recognition technology,” *MONTANA STATE LEGISLATURE*, 2021.

[6] U. Wolfe-Rocca, “Cointelpro: Teaching the fbi’s war on the black freedom movement,” *rethinking schools*, 2016.

[7] N. F. Wessler, “Police say a simple warning will prevent face recognition wrongful arrests. that’s just not true,” *American Civil Liberties Union*, 2024.

[8] K. Simon, “Protecting civil rights organizations and activists: A policy addressing the government’s use of surveillance tools,” *Federation of American Scientists*, 2023.

[9] Z. Kanno-Youngs, “U.s. watched george floyd protests in 15 cities using aerial surveillance.” *New York Times*, 2020.

[10] L. I. Smith, “A tutorial on principal components analysis,” Department of Computer Science, University of Otago, Tech. Rep., 2002.

[11] T. Evgeniou and M. Pontil, “Support vector machines: Theory and applications,” *Proceedings of the Advanced Course on Artificial Intelligence (ACAI)*, 2002.

[12] “Face recognition algorithm using eigenfaces,” https://www.youtube.com/watch?v=61NuFlK5VdUt=264s, 2021.

[13] MatrixCalc.org, “Online matrix calculator,” 2023, accessed: 2024-04-27. [Online]. Available: https://matrixcalc.org/vectors.html