Face Recognition Using Eigenface with Naive Bayes

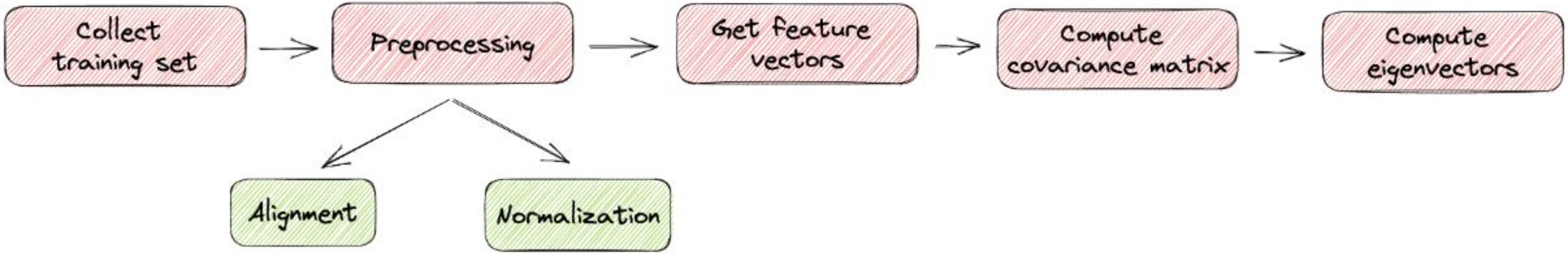
Submitted by
Harsirat Grewal and Vani Modem (Team 9)

Instructed by: Professor Sunny Raj

INTRODUCTION

- Face recognition: A biometric method based on unique facial features
- Common applications: Security systems, attendance tracking, identity verification
- Challenge: Eigenface method alone may result in low accuracy
- Proposed Solution: Combine Eigenface + Naive Bayes + Z-Score normalization

How Do Eigenfaces Work?



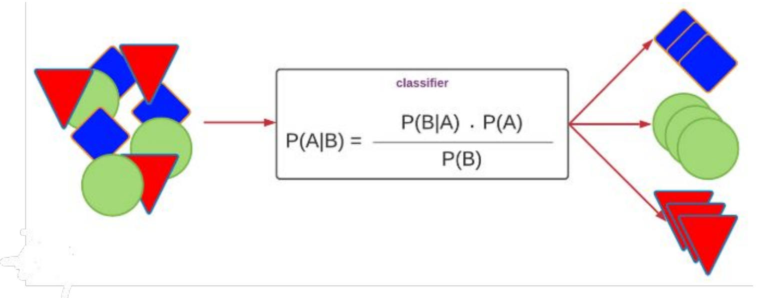
Based on PCA (Principal Component Analysis)

The goal of the method is to represent an image that depicts the face of a person as a linear combination of a set of basic images that are called eigenfaces.

Naive Bayes Classification

Naive Bayes classification is based on Bayes' theorem assuming feature independence

- For each class C, calculate the prior probability
- For each feature x_i and class C, estimate $P(x_i | C)$
- Given a new input x, calculate the posterior probability for each class
- Class with the highest probability wins => Predicted Class



$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

using Bayesian probability terminology, the above equation can be written as

$$\text{Posterior} = \frac{\text{prior x likelihood}}{\text{evidence}}$$

Z-Score Normalization

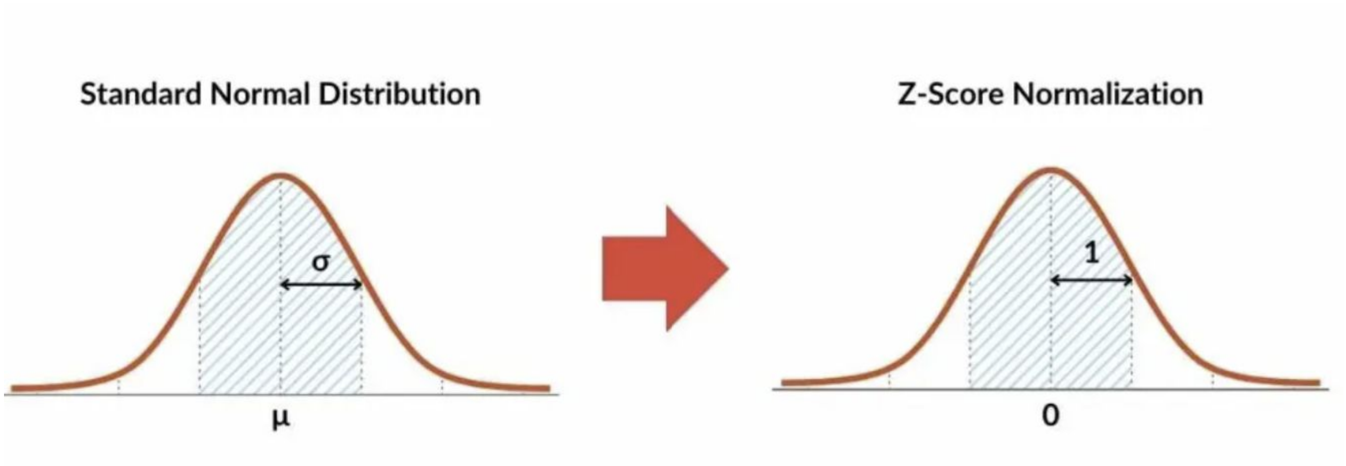
Z-score scales features to have mean = 0 and standard deviation = 1

x = individual data point

μ = mean of the dataset

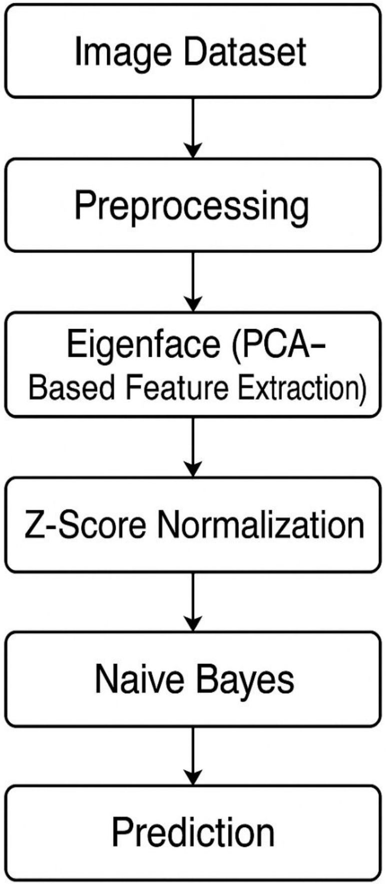
σ = standard deviation of the dataset

$$Z = \frac{X - \mu}{\sigma}$$



Implementation

- Image Preparation -> Grayscale conversion
- Vectorization -> Flatten each image into a 1D vector
- Mean Normalization - Normalize each image by subtracting the mean face
- Computer the covariance matrix -> Calculate the eigenvalues and eigenvectors
- Dimensionality Reduction Select top *k*-eigenfaces based on highest eigenvalues
- Project each face image onto this lower-dimensional eigenface space (PCA)
- Z-Score Normalization -> Standardize features using Z-Score
- Use projected and normalized features to train a Naive Bayes classifier
- For each class BBB, calculate prior probabilities P(B) and conditional probabilities P(x_i/B)



Dataset used for initial implementation

Dataset Used: ORL (Olivetti Research Laboratory) Face Database.

Contents: 400 images of 40 individuals (10 images per individual) with variations in expression and pose.

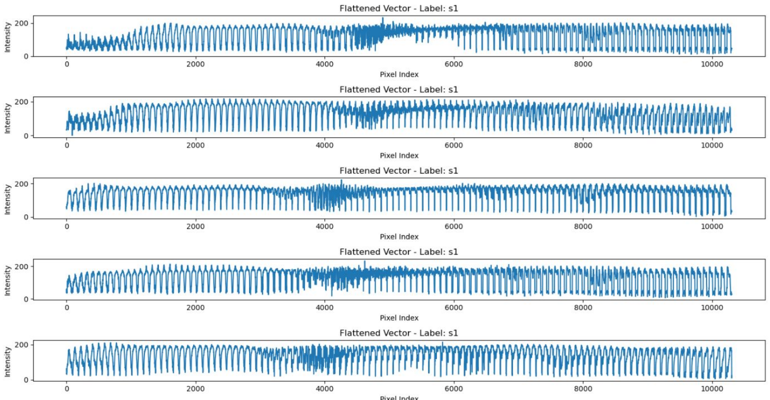
Image Specifications: Grayscale, 92x112 pixels

Gray Scale Images

```python
# Convert to grayscale
with z.open(file_info.filename) as file:
    image = Image.open(file).convert("L")
    images.append(np.array(image, dtype=np.float32))
    labels.append(label)
```

Flatten and Visualize the Image Matrix

```python
# Convert images into a matrix where each row is a flattened image vector
X = np.array([img.flatten() for img in images])
y = np.array(labels)
```

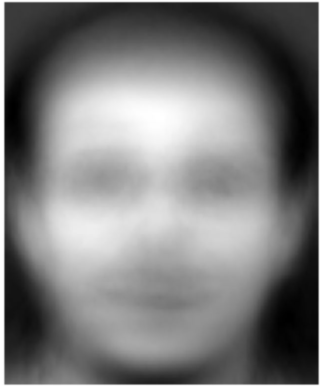


Average Face Vector

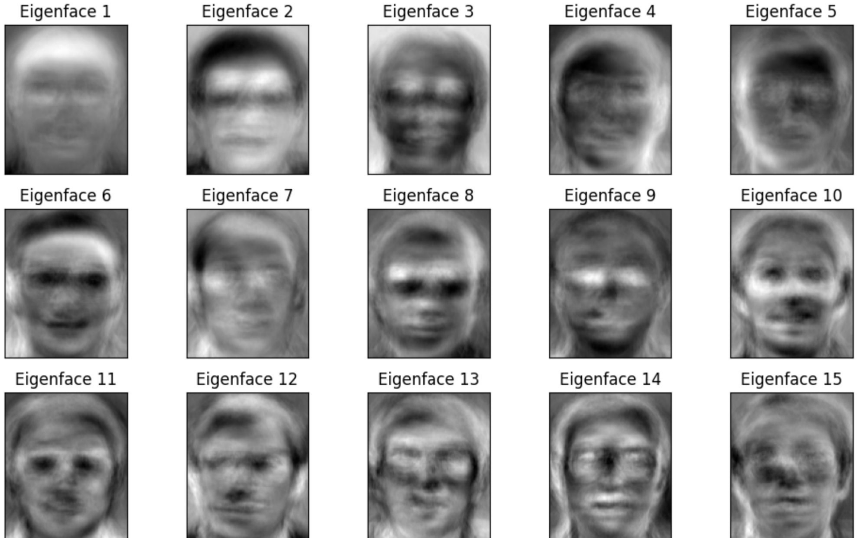
- Used to center the data (subtract from each image) before PCA
- Top eigenvectors are the eigenfaces
- Compute the covariance matrix of this centered data
- Perform Eigen decomposition and sort eigenvalues in descending order

$$Y = \frac{1}{M}\sum_{i=1}^{M} r_i$$

Average Face



```python
# Convert labels to numerical values
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# Compute the mean vector (average face)
mean_face = np.mean(X, axis=0)

# Subtract the mean from the dataset
X_centered = X - mean_face

# Compute the covariance matrix
cov_matrix = np.cov(X_centered, rowvar=False)

# Compute eigenvalues and eigenvectors
eigenvalues, eigenvectors = np.linalg.eigh(cov_matrix)

# Sort eigenvalues & eigenvectors in descending order
sorted_indices = np.argsort(eigenvalues)[::-1]
eigenvalues = eigenvalues[sorted_indices]
eigenvectors = eigenvectors[:, sorted_indices]
```



Z-Score Normalization

- Standardizes the PCA feature vectors so they are comparable
- Helps improve classification accuracy
- Makes Naive Bayes more robust by aligning features with Gaussian assumptions

```
# Normalize eigenvectors using Z-score normalization
scaler = StandardScaler()
eigenvectors_normalized = scaler.fit_transform(eigenvectors)
```

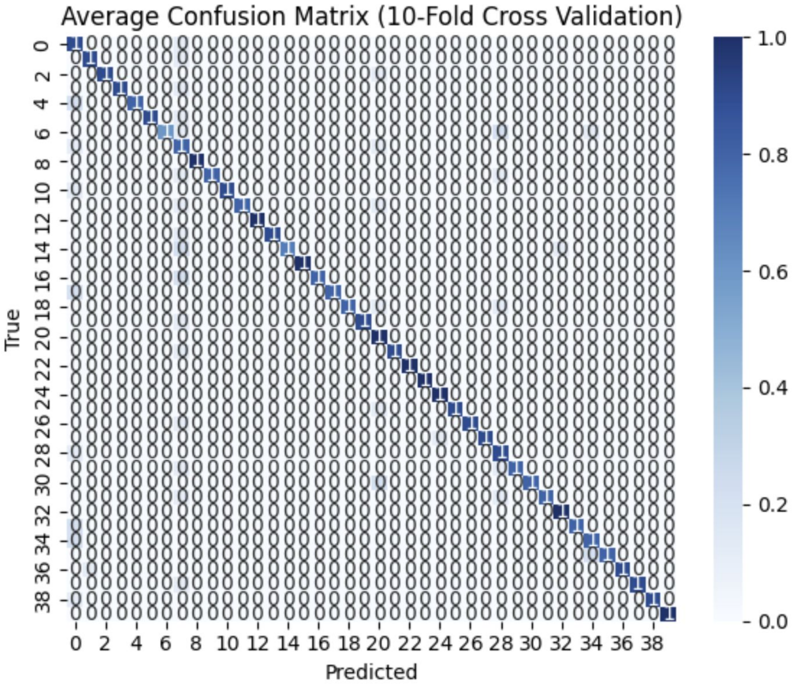
```
# Reduce dimensionality using PCA (keeping top components)
n_components = 100  # No. of principal components to keep
pca = PCA(n_components=n_components, whiten=True, random_state=42)
X_pca = pca.fit_transform(X_centered)
```

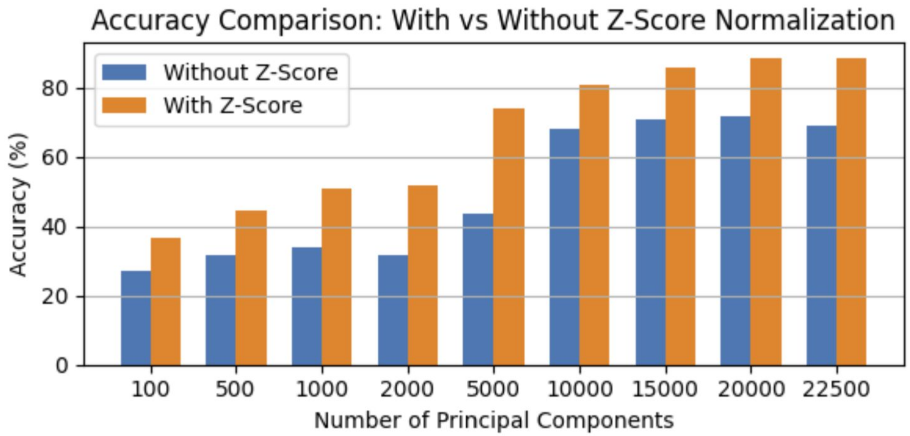
10-fold Cross Validation

- The dataset is split into 10 equal parts
- The model is trained on 9 folds and tested on 1 fold
- This process is repeated 10 times
- Reduces bias and variation in the model

```python
# 10-fold Cross Validation
kf = KFold(n_splits=10, shuffle=True, random_state=42)
conf_matrices = []
reports = []

for train_index, test_index in kf.split(X_pca):
    X_train, X_test = X_pca[train_index], X_pca[test_index]
    y_train, y_test = y_encoded[train_index], y_encoded[test_index]

    # Compute standard deviation & mean for training data
    train_mean = np.mean(X_train, axis=0)
    train_std = np.std(X_train, axis=0)

    # Compute mean for test data
    test_mean = np.mean(X_test, axis=0)

    # Train Naive Bayes classifier
    clf = GaussianNB()
    clf.fit(X_train, y_train)

    # Confusion matrix
    y_pred = clf.predict(X_test)
    conf_matrices.append(confusion_matrix(y_test, y_pred, labels=np.unique(y_encoded)))
    reports.append(classification_report(y_test, y_pred, zero_division=0))
```

Results - ORL Faces





Accuracy Comparison Table:

Principal Components	Accuracy Without Z-Score (%)	Accuracy With Z-Score (%)
100	27.0	36.5
500	31.5	44.5
1000	34.0	51.0
2000	31.5	51.5
5000	43.5	74.0
10000	68.0	81.0
15000	71.0	86.0
20000	71.5	88.5
22500	69.0	88.5

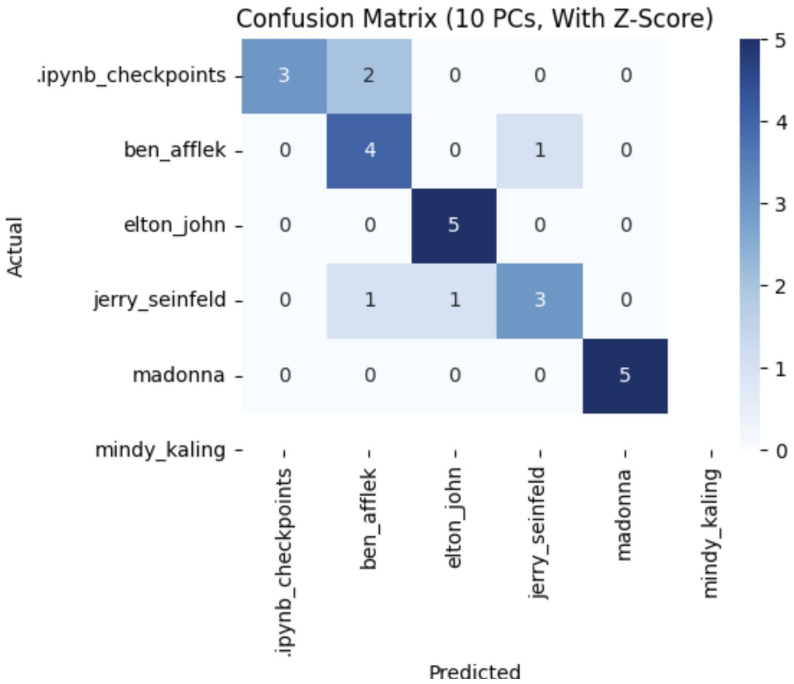
Dataset used for testing the implementation

5 Celebrities Dataset
Ben Affleck, Elton John, Jerry Seinfeld, Madonna, Mindy Kaling

Training

Validation

Results - Testing (5 Celebrities)



Accuracy Comparison Table:

Principal Components	Accuracy Without Z-Score (%)	Accuracy With Z-Score (%)
10	76.0	76.0
20	72.0	72.0
30	76.0	76.0
40	68.0	68.0
50	72.0	72.0
60	68.0	68.0
70	56.0	56.0
80	56.0	56.0
90	40.0	40.0

Eigenface Advantages

Eigenface provides an easy and cheap way to realize face recognition in that:

- Its training process is completely automatic and easy to code.
- Eigenface adequately reduces statistical complexity in face image representation.
- Once eigenfaces of a database are calculated, face recognition can be achieved in real time.
- Eigenface can handle large databases.
- Combining eigenface with Naive Bayes classification in face recognition offers advantages like simplified feature extraction and reduced computational cost.
- Eigenface uses Principal Component Analysis (PCA) to represent faces, while Naive Bayes provides a probabilistic classification method. This combination can improve accuracy and reduce the need for extensive training data.

Eigenface Disadvantages

- It is very sensitive to lighting, scale and translation, and requires a highly controlled environment.
- Eigenface has difficulty capturing expression changes.
- The most significant eigenfaces are mainly about illumination encoding and do not provide useful information regarding the actual face.