BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN - TIN HỌC
BỘ MÔN PHƯƠNG PHÁP SỐ TRONG KHOA HỌC DỮ LIỆU

BÁO CÁO ĐỒ ÁN

ỨNG DỤNG PCA TRONG NHẬN DIỆN KHUÔN MẶT

NHÓM 2

Họ và tên	**MSSV**
Lê Trọng Nghĩa	22280059
Võ Duy Nghĩa	22280060
Nguyễn Hồ Nam	22280057
Phạm Tấn Phước	22280069
Nguyễn Thiên Phúc	22280067
Dương Thanh Phong	22280063
Lê Viết Tố Khoa	22280046

Tháng 7/2025

Tóm tắt chủ đề

Nhận diện khuôn mặt là một lĩnh vực quan trọng trong thị giác máy tính và trí tuệ nhân tạo, với nhiều phương pháp đã được phát triển nhằm tăng độ chính xác và hiệu quả. Báo cáo này trình bày về phương pháp Phân tích thành phần chính (PCA), một kỹ thuật giảm chiều dữ liệu phổ biến, giúp trích xuất đặc trưng quan trọng từ ảnh khuôn mặt. PCA không chỉ giúp giảm độ phức tạp tính toán mà còn cải thiện khả năng phân loại trong các hệ thống nhận diện khuôn mặt. PCA được dùng chung với các model khác để nhận diện khuôn mặt của các thành viên trong nhóm 2. Các hạn chế về PCA cũng được nói đến ở cuối bài báo cáo này.

Mục lục

1 Giới thiệu khái quát về PCA

PCA (Principal Component Analysis - Phân tích thành phần chính) là một kỹ thuật dùng để giảm số chiều của dữ liệu. Mục tiêu của PCA là giữ lại những thông tin quan trọng nhất từ dữ liệu gốc trong khi loại bỏ những phần nhiễu hoặc ít quan trọng. Kết quả là một tập hợp mới gồm các biến (gọi là các thành phần chính), không còn tương quan với nhau và mang ý nghĩa đơn giản hơn.

PCA làm gì?

- Giảm số lượng thông tin (giảm chiều) để dễ xử lý, tiết kiệm thời gian và bộ nhớ.
- Loại bỏ những phần "nhiễu"(thông tin không quan trọng).
- Biến dữ liệu phức tạp thành dạng dễ hiểu hơn.

Ví dụ thực tế:

- **Xử lý ảnh:** PCA có thể giảm hàng nghìn điểm ảnh trong một bức ảnh khuôn mặt xuống còn vài đặc điểm chính (như hình dạng mắt, mũi, miệng), giúp máy tính nhận diện khuôn mặt nhanh hơn.
- **Kinh tế:** PCA rút gọn hàng trăm chỉ số kinh tế (như GDP, lãi suất, lạm phát) thành vài yếu tố chính để dự đoán xu hướng thị trường.

PCA được Karl Pearson phát minh năm 1901, và sau đó được Harold Hotelling phát triển thêm vào những năm 1930. Nó giống như việc tìm ra "góc nhìn tốt nhất"để quan sát dữ liệu!

2 Các khái niệm thống kê cơ bản

Trung bình (Kỳ vọng, Mean)

Trung bình cộng của một tập dữ liệu. Cho N giá trị $x_1, x_2, x_3, \ldots, x_N$:

$$\bar{X} = \frac{x_1 + x_2 + x_3 + \cdots + x_N}{N} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Phương sai (Variance)

Đo độ phân tán của dữ liệu quanh giá trị trung bình. Phương sai càng nhỏ thì các điểm dữ liệu càng gần với kỳ vọng (giá trị trung bình), tức các điểm dữ liệu càng giống nhau.

$$\sigma^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

Hiệp phương sai (Covariance)

Đo mức độ hai biến thay đổi cùng nhau. Nếu hai biến tăng/giảm cùng chiều, hiệp phương sai dương; ngược chiều thì âm.

$$\text{cov}(X, Y) = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})(y_i - \bar{y})$$

Ví dụ:

- Nếu bạn cao hơn thì thường nặng hơn → hiệp phương sai dương (cùng tăng).
- Nếu bạn học nhiều hơn thì thời gian chơi game ít đi → hiệp phương sai âm (ngược chiều).

Ma trận hiệp phương sai

Là một ma trận vuông biểu diễn mối quan hệ giữa nhiều biến. Mỗi phần tử (i, j) là hiệp phương sai giữa biến i và biến j. Ma trận hiệp phương sai là ma trận đối xứng. Cho N điểm dữ liệu được biểu diễn bởi các vector cột $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ khi đó, vector kỳ vọng và ma trận hiệp phương sai của toàn bộ dữ liệu được định nghĩa là:

Vector kỳ vọng:

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i$$

Ma trận hiệp phương sai:

$$S = \frac{1}{N-1} \sum_{i=1}^{N} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T$$

3 Trị riêng và vector riêng

Trong phương pháp Phân tích Thành phần Chính (Principal Component Analysis - PCA), mục tiêu là tìm ra các *hướng* trong không gian dữ liệu mà tại đó dữ liệu biến đổi nhiều nhất. Các hướng này chính là các **vector riêng** (eigenvectors) của ma trận hiệp phương sai, còn độ lớn của sự biến đổi theo các hướng đó được định lượng bằng các **trị riêng** (eigenvalues).

3.1 Trị riêng (Eigenvalue)

Trị riêng là một đại lượng vô hướng (số thực) biểu thị mức độ thay đổi (phân tán) của dữ liệu theo một hướng cụ thể trong không gian. Một trị riêng càng lớn thì hướng tương ứng càng quan trọng trong việc mô tả dữ liệu – vì dữ liệu trải rộng nhiều theo hướng đó. Trong PCA, các trị riêng giúp xác định mức độ "đóng góp"của từng thành phần chính vào tổng phương sai của toàn bộ dữ liệu.

Cách tính trị riêng: Để tìm trị riêng, ta cần giải phương trình đặc trưng của ma trận hiệp phương sai S:

$$\det(S - \lambda I) = 0$$

Trong đó:

- S: ma trận hiệp phương sai của dữ liệu ($n \times n$),
- λ: trị riêng cần tìm,
- I: ma trận đơn vị cùng kích thước với S,
- det: ký hiệu định thức của ma trận.

Phương trình này là một đa thức bậc n theo biến λ. Nghiệm của phương trình chính là các trị riêng của ma trận S.

3.2 Vector riêng (Eigenvector)

Vector riêng là những vector không thay đổi hướng khi bị tác động bởi một ma trận — chỉ thay đổi về độ dài. Cụ thể, nếu u là vector riêng và λ là trị riêng tương ứng, thì:

$$Su = \lambda u$$

hay viết lại thành phương trình thuần nhất:

$$(S - \lambda I)u = 0$$

Đây là hệ phương trình tuyến tính đồng nhất, và nghiệm của hệ là các vector riêng u tương ứng với trị riêng λ.

Trong PCA: Các vector riêng xác định các hướng chính mà dữ liệu thay đổi nhiều nhất. Vector riêng tương ứng với trị riêng lớn nhất là thành phần chính đầu tiên, đại diện cho phương sai lớn nhất trong dữ liệu. Các vector riêng tương ứng với các trị riêng nhỏ hơn lần lượt là các thành phần chính tiếp theo. Tập hợp các vector riêng (được chuẩn hóa và sắp xếp theo trị riêng giảm dần) tạo thành cơ sở mới của không gian dữ liệu, cho phép ta chiếu dữ liệu về không gian có số chiều thấp hơn nhưng vẫn giữ lại thông tin quan trọng nhất.

4 Phân tích Thành phần Chính (PCA)

Ý tưởng cốt lõi

Thuật toán PCA tìm một hệ không gian mới và tối đa hóa phương sai dữ liệu của không gian mới đó. Sau đó lựa chọn ra n chiều có phương sai lớn nhất (giả thuyết rằng dữ liệu càng phân tán, phương sai càng lớn thì càng có giá trị).

Mục tiêu của PCA

PCA (Principal Component Analysis) là một kỹ thuật giảm chiều dữ liệu. Mục tiêu là tìm một không gian con có số chiều thấp hơn để biểu diễn dữ liệu gốc sao cho phương sai (variance) của dữ liệu được chiếu lên không gian đó là lớn nhất. Việc tối đa hóa phương sai giúp bảo toàn tối đa thông tin về sự khác biệt giữa các điểm dữ liệu.

Giả sử ta có một tập dữ liệu X gồm m mẫu, mỗi mẫu có n đặc trưng.

$$X \in \mathbb{R}^{m \times n}$$

Bước 1: Chuẩn hóa dữ liệu (Standardization)

Lý do

PCA rất nhạy cảm với thang đo của các đặc trưng. Các đặc trưng có thang đo lớn sẽ lấn át và ảnh hưởng đến kết quả.

Thực hiện

Với mỗi đặc trưng (cột) thứ j của dữ liệu, ta chuẩn hóa nó bằng cách trừ đi giá trị trung bình μ_j và chia cho độ lệch chuẩn σ_j.

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}$$

Trong đó x_{ij} là giá trị của mẫu thứ i ở đặc trưng thứ j. Sau khi chuẩn hóa, mỗi đặc trưng sẽ có trung bình bằng 0 và độ lệch chuẩn bằng 1. Gọi ma trận dữ liệu đã chuẩn hóa là Z.

Bước 2: Tính ma trận Hiệp phương sai (Covariance Matrix)

Lý do

Ma trận hiệp phương sai cho biết mối quan hệ tuyến tính và sự biến đổi đồng thời giữa các cặp đặc trưng.

- $\text{Cov}(X_j, X_k) > 0$: Đồng biến.
- $\text{Cov}(X_j, X_k) < 0$: Nghịch biến.
- $\text{Cov}(X_j, X_k) \approx 0$: Không có quan hệ tuyến tính.

Thực hiện

Ma trận hiệp phương sai Σ (kích thước $n \times n$) của dữ liệu đã chuẩn hóa Z được tính như sau:

$$\Sigma = \frac{1}{m-1} Z^T Z$$

Bước 3: Phân rã Eigen (Eigen-decomposition)

Đây là bước cốt lõi, ta tìm các vector riêng (eigenvectors) và giá trị riêng (eigenvalues) của ma trận hiệp phương sai Σ.

$$\Sigma \mathbf{v} = \lambda \mathbf{v}$$

- **Vector riêng ($\mathbf{v}$)**: Là các vector chỉ phương của các trục mới (thành phần chính). Chúng tạo thành một hệ cơ sở trực giao.
- **Giá trị riêng (λ)**: Cho biết lượng phương sai của dữ liệu được giải thích bởi vector riêng tương ứng. λ càng lớn, thành phần chính càng quan trọng.

Bước 4: Sắp xếp và Chọn các Thành phần chính

Sắp xếp các cặp (giá trị riêng, vector riêng) theo thứ tự giá trị riêng giảm dần:

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$$

Vector riêng $\mathbf{v}_1$ tương ứng với λ_1 là Thành phần chính thứ nhất (PC1), $\mathbf{v}_2$ là PC2, v.v.

Ta chọn k thành phần chính đầu tiên ($k < n$) dựa trên **Tỷ lệ phương sai giải thích được**:

$$\text{Explained Variance Ratio} = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{j=1}^{n} \lambda_j}$$

Thông thường, ta chọn k sao cho tỷ lệ này đạt một ngưỡng nhất định (ví dụ: 95%).

Bước 5: Chiếu dữ liệu lên không gian mới

Tạo ma trận W (kích thước $n \times k$) chứa k vector riêng đã chọn làm các cột.

$$W = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k]$$

Dữ liệu mới Y (kích thước $m \times k$) thu được bằng cách chiếu dữ liệu đã chuẩn hóa Z lên W:

$$Y = ZW$$

5 PCA trong nhận diên khuôn mặt

Ý tưởng Cốt lõi

Mỗi khuôn mặt có thể được biểu diễn như một tổ hợp tuyến tính của một "khuôn mặt trung bình"và các "đặc trưng khuôn mặt"(Eigenfaces).

- **Khuôn mặt trung bình (Ψ)**: Đặc điểm chung của tất cả các khuôn mặt.
- **Eigenfaces ($\mathbf{u}_i$)**: Các thành phần chính biểu diễn các biến thể chính giữa các khuôn mặt.

Một khuôn mặt bất kỳ Γ có thể được tái tạo lại xấp xỉ bằng công thức:

$$\Gamma \approx \Psi + \sum_{i=1}^{k} w_i \mathbf{u}_i$$

Vector trọng số $\mathbf{\Omega} = [w_1, w_2, \ldots, w_k]^T$ chính là "mã nhận dạng"của khuôn mặt Γ.

Bước 1 & 2: Chuẩn bị Dữ liệu và Vector hóa

Thu thập một bộ M ảnh khuôn mặt huấn luyện $\{\Gamma_1, \Gamma_2, \ldots, \Gamma_M\}$. Sau khi tiền xử lý (căn chỉnh, cắt, chuẩn hóa kích thước thành $W \times H$ pixels), mỗi ảnh 2D Γ_i được "duỗi thẳng"thành một vector cột 1D có $N = W \times H$ chiều.

Bước 3: Tính Khuôn mặt Trung bình (Ψ)

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

Bước 4: Tính các Vector Sai khác (Φ_i)

Đây là bước chuẩn hóa về trung bình 0.

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

Bước 5: Áp dụng PCA - Tìm các Eigenfaces

1. **Xây dựng Ma trận Dữ liệu A**: Tạo ma trận A (kích thước $N \times M$) với các cột là các vector sai khác.
$$A = [\Phi_1, \Phi_2, \ldots, \Phi_M]$$

2. **Tính Ma trận Hiệp phương sai C**:
$$C = AA^T \quad \text{(kích thước } N \times N, \text{ rất lớn)}$$

3. **Sử dụng "Thủ thuật PCA"**: Thay vì làm việc với C, ta tính toán trên ma trận L nhỏ hơn.
$$L = A^T A \quad \text{(kích thước } M \times M)$$
Tìm giá trị riêng λ_i và vector riêng $\mathbf{v}_i$ của L bằng cách giải:
$$L\mathbf{v}_i = \lambda_i \mathbf{v}_i$$

4. **Tính các Eigenfaces $\mathbf{u}_i$**: Các vector riêng của C (chính là Eigenfaces) được tính từ các vector riêng của L.
$$\mathbf{u}_i = A\mathbf{v}_i$$

5. **Chuẩn hóa, Sắp xếp và Chọn**: Chuẩn hóa các $\mathbf{u}_i$ (e.g., $\|\mathbf{u}_i\| = 1$), sắp xếp chúng theo λ_i giảm dần, và chọn ra k Eigenfaces tốt nhất ($k \ll M$) để tạo thành một hệ cơ sở cho **Không gian mặt (Face Space)**.

Bước 6: Chiếu Dữ liệu Huấn luyện vào Không gian Mặt

Với mỗi vector sai khác Φ_i, ta tính vector trọng số tương ứng Ω_i (kích thước $k \times 1$).

$$\Omega_i = \begin{pmatrix} w_{1i} \\ w_{2i} \\ \vdots \\ w_{ki} \end{pmatrix}, \quad \text{với} \quad w_{ji} = \mathbf{u}_j^T \Phi_i$$

Vector Ω_i là biểu diễn của khuôn mặt Γ_i trong không gian mặt. Ta lưu lại các cặp $(\Omega_i, \text{Nhãn}_i)$.

Bước 7: Nhận diện bằng 1-NN

Sau khi đã có tập các cặp $(\Omega_i, \text{Nhãn}_i)$ từ dữ liệu huấn luyện, ta thực hiện nhận diện khuôn mặt mới Γ_{test} theo các bước:

1. Chuẩn hóa ảnh đầu vào: $\Phi_{\text{test}} = \Gamma_{\text{test}} - \Psi$
2. Chiếu vào không gian mặt:

$$\Omega_{\text{test}} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \end{pmatrix}, \quad \text{với} \quad w_j = \mathbf{u}_j^T \Phi_{\text{test}}$$

3. Tính khoảng cách Euclid đến từng vector huấn luyện:

$$d_i = \|\Omega_{\text{test}} - \Omega_i\|_2$$

4. Dự đoán nhãn bằng cách chọn vector gần nhất:

$$i^* = \arg\min_i d_i$$

5. Nếu $d_{i^*} < \theta$, ta gán nhãn của ảnh test là Nhãn_{i^*}. Ngược lại, kết luận đây là một khuôn mặt **lạ** (unknown).

Đây chính là thuật toán **1-Nearest Neighbor (1-NN)** áp dụng trong không gian đặc trưng PCA. Trong thuật toán nhóm có sử dụng nhiều giá trị K khác nhau cho thuật toán KNN để so sánh và đưa ra giá trị tối ưu nhất.

6 Một số tính chất toán học quan trọng của PCA

Ma trận hiệp phương sai luôn đối xứng và bán xác định dương (mọi trị riêng ≥ 0)

Một tính chất quan trọng của ma trận hiệp phương sai là nó luôn đối xứng. Điều này xuất phát từ cách tính toán hiệp phương sai: hiệp phương sai giữa biến X_i và X_j luôn bằng hiệp phương sai giữa X_j và X_i, tức là $\text{Cov}(X_i, X_j) = \text{Cov}(X_j, X_i)$. Do đó, các phần tử trên đường chéo chính của ma trận hiệp phương sai thể hiện phương sai của từng biến, trong khi các phần tử ngoài đường chéo thể hiện hiệp phương sai giữa các cặp biến.

Ngoài ra, ma trận hiệp phương sai còn có tính chất bán xác định dương, nghĩa là tất cả các trị riêng (eigenvalues) của ma trận này đều lớn hơn hoặc bằng 0. Tính chất này đảm bảo rằng phương sai của dữ liệu (được biểu diễn qua các trị riêng) không bao giờ âm, phù hợp với ý nghĩa vật lý của phương sai trong thống kê. Tính bán xác định dương cũng đóng vai trò quan trọng trong việc đảm bảo rằng các phép biến đổi của PCA luôn giữ được tính ổn định và ý nghĩa thống kê.

Các vector riêng tạo thành một hệ trực giao, nghĩa là chúng độc lập (không liên quan tới nhau)

Một kết quả quan trọng khác của PCA là các vector riêng (eigenvectors) của ma trận hiệp phương sai tạo thành một hệ trực giao. Điều này có nghĩa là các vector riêng vuông góc với nhau trong không gian Euclidean, và tích vô hướng giữa bất kỳ hai vector riêng nào khác nhau đều bằng 0. Hơn nữa, các vector riêng thường được chuẩn hóa để có độ dài bằng 1, hình thành một hệ cơ sở trực chuẩn.

Tính trực giao của các vector riêng đảm bảo rằng các thành phần chính (principal components) – tức là các hướng mới mà PCA tìm ra – là độc lập tuyến tính với nhau. Điều này rất quan trọng trong việc giảm chiều dữ liệu, vì nó loại bỏ sự dư thừa thông tin giữa các thành phần chính, giúp biểu diễn dữ liệu một cách hiệu quả và không bị trùng lặp.

Tổng các trị riêng bằng tổng phương sai của các biến ban đầu

Một tính chất toán học thú vị của PCA là tổng các trị riêng của ma trận hiệp phương sai bằng tổng phương sai của các biến ban đầu trong tập dữ liệu. Cụ thể, nếu ta có p biến gốc, phương sai của mỗi biến được biểu diễn trên đường chéo chính của ma trận hiệp phương sai, và tổng các phương sai này chính là dấu vết (trace) của ma trận. Đồng thời, tổng các trị riêng của ma trận hiệp phương sai cũng bằng dấu vết này.

$$\sum_{i=1}^{p} \lambda_i = \text{trace}(S) = \sum_{j=1}^{p} \text{Var}(X_j)$$

Tính chất này có ý nghĩa thống kê quan trọng: nó cho thấy rằng toàn bộ sự biến thiên (phương sai) của dữ liệu gốc được bảo toàn trong quá trình phân tích PCA. Các trị riêng đại diện cho lượng phương sai mà mỗi thành phần chính giải thích, và tổng các trị riêng phản ánh toàn bộ thông tin phương sai của tập dữ liệu ban đầu.

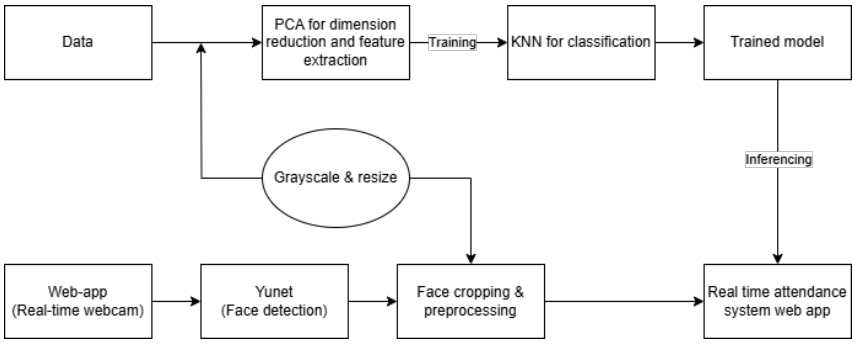
PCA tìm được trục tọa độ mới sao cho dữ liệu có phương sai lớn nhất trên các trục này

Mục tiêu chính của PCA là tìm ra một hệ tọa độ mới (các thành phần chính) sao cho dữ liệu được biểu diễn với phương sai lớn nhất trên các trục này. Các thành phần chính được chọn lần lượt theo thứ tự giảm dần của trị riêng tương ứng, nghĩa là thành phần chính đầu tiên sẽ nằm dọc theo hướng có phương sai lớn nhất, thành phần chính thứ hai nằm dọc theo hướng có phương sai lớn thứ hai (và vuông góc với thành phần thứ nhất), và cứ tiếp tục như vậy.

Quá trình này đảm bảo rằng phần lớn sự biến thiên trong dữ liệu được giữ lại chỉ với một số ít thành phần chính, giúp giảm chiều dữ liệu mà vẫn duy trì được thông tin quan trọng. Các trục tọa độ mới này không chỉ trực giao mà còn được tối ưu hóa để biểu diễn dữ liệu một cách hiệu quả nhất, làm nền tảng cho các ứng dụng như nén dữ liệu, trực quan hóa, hay xử lý tín hiệu.

7 Thiết kế và cài đặt hệ thống nhận diện khuôn mặt bằng PCA

7.1 Sơ đồ mô tả hệ thống



Hình 1: Diagram mô tả pipeline của ứng dụng

7.2 Các thành phần chính

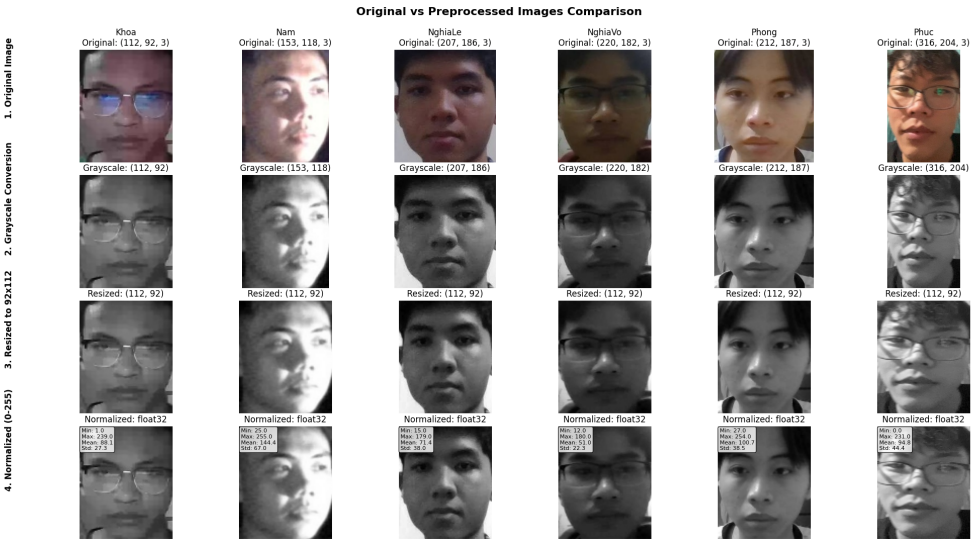
7.2.1 Tiền xử lý dữ liệu ảnh

Dữ liệu gốc

Nhóm đã thu thập 10 tấm ảnh của khuôn mặt cho từng người trong nhóm với tổng cộng là 70 tấm. Các tấm ảnh được chụp khác nhau với một số khác biệt như về ánh sáng, góc độ, độ gần giữa mặt và máy ảnh, yếu tố nền.

Tuy nhiên, trong lúc huấn luyện mô hình và dự đoán thì kết quả ra không như nhóm mong đợi, có thể phần lớn do PCA khá dễ bị các giá trị ngoại lai (outlier) làm ảnh hưởng đến khả năng phân loại. Do đó, nhóm quyết định là dùng Yunet và webcam để trích xuất

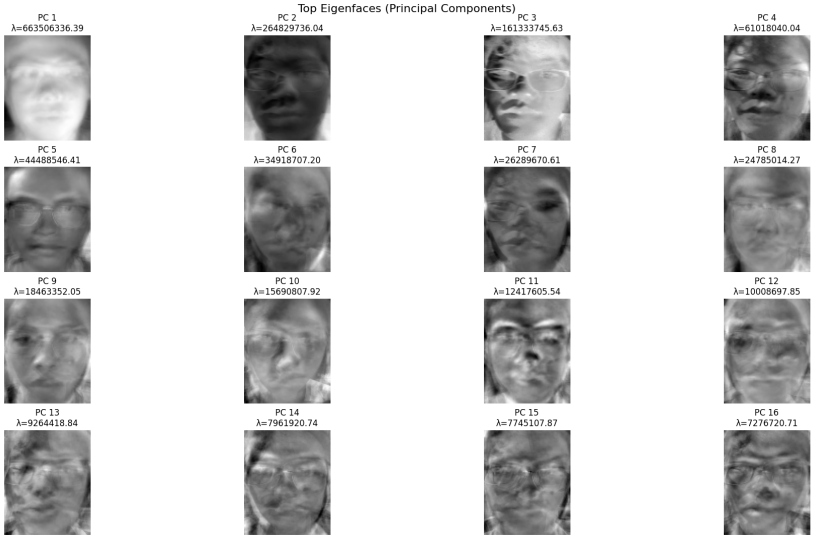
dữ liệu trực tiếp với số lượng dữ liệu nhiều hơn đáng kể.



Hình 2: Các khuôn mặt trước và sau khi preprocess

Sau khi có được dữ liệu là một tấm ảnh gốc có khuôn mặt. Nhóm đã chuyển tấm ảnh về ảnh trắng đen và dùng Yunet để tìm khuôn mặt trong tấm ảnh và cắt ra thành một hình chữ nhật 112x92 pixel.

7.2.2 Trích xuất đặc trưng bằng PCA



Hình 3: Các Eigenfaces với mỗi PC khác nhau

7.2.3 Thuật toán viết bằng ngôn ngữ Python

```
def find_k(eigenvalues, quality_percent):
    """
```

```
        Find the optimal number of PC for the quality
           percentage.

        Args:
            eigenvalues (np.ndarray): Descending eigenvalues
               array.
            quality_percent (float): Quality percentage
               retained.

        Returns:
            int: Number of components
        """
        total_energy = np.sum(eigenvalues)
        threshold = quality_percent / 100 * total_energy
        temp = 0
        K = 0
        while temp < threshold:
            temp += eigenvalues[K]
            K += 1
        return K
```

Listing 1: Hàm tìm số lượng thành phần chính K

```
    def compute_auto_threshold(self, method='percentile',
       value=95):
    dists = []
    for label in np.unique(self.y):
        indices = self.label_indices[label]
        mean_vec = self.class_means[label]
        for idx in indices:
            sample = self.new_coordinates[:, idx]
            dist = np.linalg.norm(sample - mean_vec)
            dists.append(dist)
    dists = np.array(dists)
    return np.percentile(dists, value)
```

Listing 2: Hàm tính ngưỡng phân loại dựa trên khoảng cách.

```
    def __init__(self, images, y, target_names, no_of_elements
       , quality_percent=100):
    """
    images: 2D Array (N^2 x M), Each column is a flattened
       image.
    y: List of labels corresponds to each person.
    target_names: Member name.
    no_of_elements: Number of images in each layer.
    quality_percent: Percentage of retained quality.
    """
    self.no_of_elements = no_of_elements
    self.images = np.asarray(images)
    self.y = y
    self.target_names = target_names
```

```
        self.quality_percent = quality_percent

        # 1. Calculate mean face
        mean = np.mean(self.images, axis=1)
        self.mean_face = mean.reshape(-1, 1)

        # 2. Standardization by subtracting the mean face
        self.training_set = self.images - self.mean_face

        # 3. Calculate eigenfaces and eigenvalues
        self.eigenfaces, self.eigenvalues = self._get_eigenfaces(
            self.training_set)

        # 4. Project new data onto a PCA dimension
        self.new_coordinates = self.get_projected_data()

        # 5. Calculate the mean vector
        self._compute_class_means()

        # 6. Calculate the threshold value automatically
        self.threshold = self.compute_auto_threshold()
```

Listing 3: Hàm khởi tạo, tính PCA và các đặc trưng cần thiết.

```
        def get_projected_data(self):
        """
        Project new face onto new dimension (w_i = U^T * (x_i -
            mean)).
        """
        Phi = self.training_set
        return self.eigenfaces.T @ Phi  # (K x M)
```

Listing 4: Chiếu dữ liệu lên PCA-space.

```
        def _compute_class_means(self):
        """
        Calculate mean vector for each class.
        """
        self.class_means = {} # {label: mean_vector}
        #Empty dictionary for storing mean_vector.
        # Key: label

        self.label_indices = {}

        # Pass through all unique labels
        for label in np.unique(self.y):

            # Return list of indices for each photo
            indices = np.where(np.array(self.y) == label)[0]

            # Store label to label_dictionary
            self.label_indices[label] = indices
```

```
        #Extract the PCA feature vectors corresponding to the
           images belonging to the label class.
        class_vectors = self.new_coordinates[:, indices]

        #Calculate the mean PCA feature vector for the images
           of the specified label class.
        self.class_means[label] = np.mean(class_vectors, axis
           =1)
```

Listing 5: Tính trung bình từng lớp trong PCA-space.

```
    def recognize_face_knn(self, face_vector, k=3, threshold=
       None):
    """
    Perform face recognition in the PCA space using a
       threshold-based method combined with KNN.
    """
    if face_vector.shape != self.mean_face.shape:
        raise ValueError("Image does not have the same
           dimensions as the training images.")
    if threshold is None:
        threshold = self.threshold

    phi = face_vector - self.mean_face
    projected = self.eigenfaces.T @ phi
    projected = projected.flatten()

    # Compute distance to the mean vector of each class
    distances_to_class = {
        label: np.linalg.norm(projected - mean_vec) for label,
            mean_vec in self.class_means.items()
    }

    # Filter classes with distance < threshold
    valid_labels = [label for label, dist in
       distances_to_class.items() if dist < threshold]

    if not valid_labels:
        return "Unknown"

    # Get all images from valid classes
    candidate_indices = []
    for label in valid_labels:
        candidate_indices.extend(self.label_indices[label])
        # self.label_indices[label] is a list of indices of
           images in the label class.
        # extend(...) adds all the image indices from that
           class to candidate_indices

    # Compute distance to each candidate image
    candidates = self.new_coordinates[:, candidate_indices].T
```

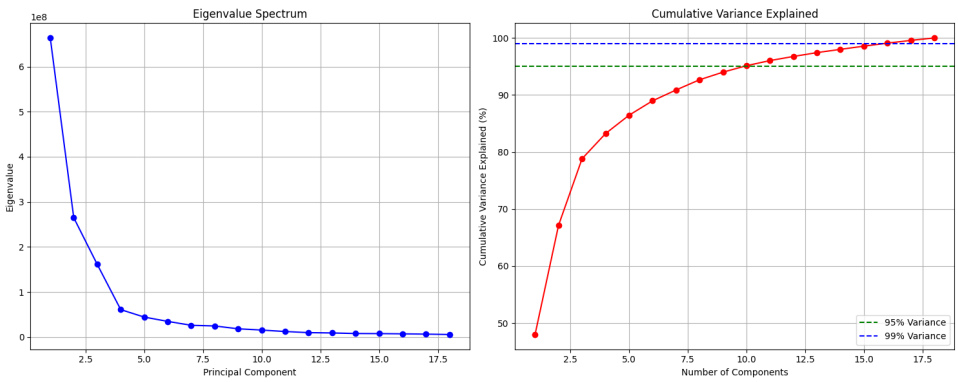
```
        dists = np.linalg.norm(candidates - projected, axis=1)

        # Get k nearest neighbors
        if len(dists) < k:
            # If number of valid images < k, reduce k to number of
                available images
            k = len(dists)

        #  Return the indices of the k closest images (sorted by
           increasing distance)
        knn_indices = np.argsort(dists)[:k]

        # Get label that is the nearest.
        knn_labels = [self.y[candidate_indices[i]] for i in
           knn_indices]

        # Voting
        vote_count = {}
        for label in knn_labels:
            vote_count[label] = vote_count.get(label, 0) + 1

        predicted_label = max(vote_count, key=vote_count.get)
        return self.target_names[predicted_label]
```

Listing 6: Nhận diện khuôn mặt bằng KNN + ngưỡng.

8 Triển khai và thực nghiệm

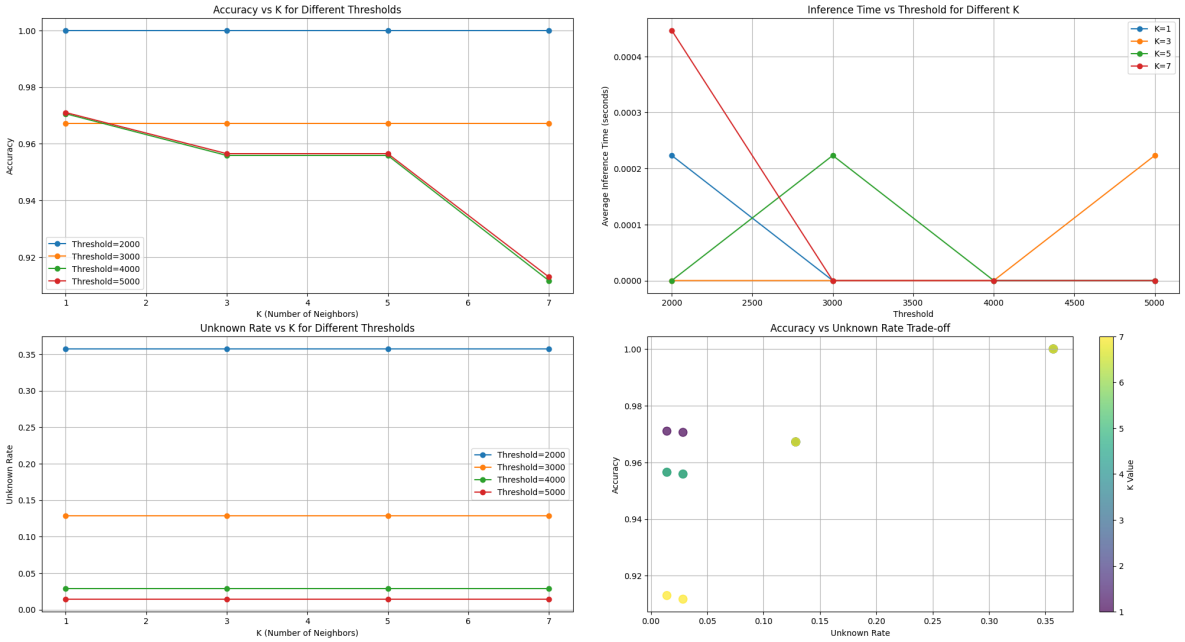
8.1 Các bước huấn luyện mô hình

Nhóm lấy mẫu ngẫu nhiên 20 tấm ảnh làm dữ liệu và 10 tấm ngẫu nhiên trong số đó làm dữ liệu huấn luyện. Các tấm ảnh được tiền xử lý với các bước đã nêu trên và được PCA. Sau đó nhóm sử dụng KNN để có thể huấn luyện dựa trên nhãn dán. Số lượng `n_components` cũng được chọn dựa trên 95% dữ liệu giữ lại (trong mô hình này, 95% nằm ở `n_components` = 10)



Hình 4: Elbow curve cho PCA

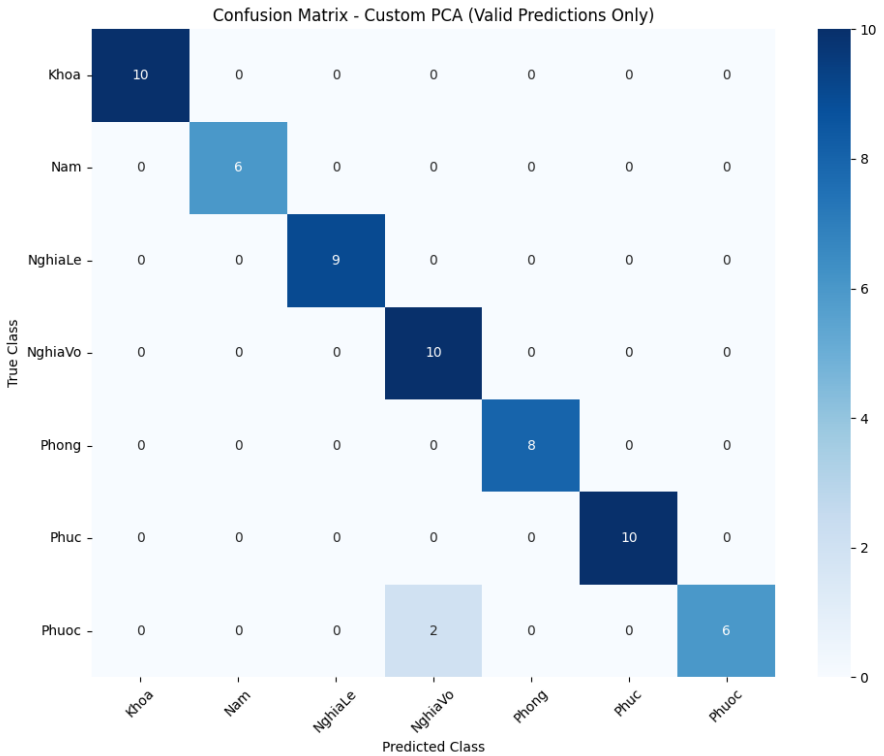
Các giá trị threshold để phân loại người bên ngoài được nhóm thí nghiệm từ 2000-5000 để đưa ra giá trị tốt nhất (Threshold được nhóm chọn trong bài là 4000)



Hình 5: Các metric cho mỗi giá trị threshold khác nhau

8.2 Kết quả thực nghiệm

Các metric sau khi huấn luyện mô hình trên 10 ảnh và test trên 10 ảnh còn lại



Hình 6: Ma trận nhầm lẫn

Nam và Phước là hai bạn mà mô hình dự đoán sai nhiều nhất. Điều này cũng dễ hiểu khi mà trong bộ dữ liệu, hình ảnh Nam và Phước có phần chá sáng và độ phân giải kém hơn so với các bạn còn lại. Ánh sáng và độ phân giải là những nhược điểm chính của PCA.

Các metric sau khi huấn luyện mô hình

```
Overall Accuracy (with unknowns): 0.8429
Valid Predictions Accuracy: 0.9672
Unknown Rate: 0.1286
Average Inference Time: 0.000412s
Training Time: 0.0092s
Total Test Images: 70
Correctly Classified: 59
Unknown Predictions: 9
```

Hình 7: Metrics

9 Chạy trực tiếp hệ thống nhận diện khuôn mặt

Ứng dụng sử dụng mô hình sau khi đã được huấn luyện để nhận diện khuôn mặt của các thành viên trong nhóm. Nhóm đã thiết kế web app cho ứng dụng trên. Ứng dụng sẽ sử dụng webcam của người dùng làm input đầu vào và cứ mỗi giây lấy một số lượng khung hình tùy chỉnh (số lượng khung hình trên giây nhóm sử dụng là 15 để tối đa giữa hiệu năng và độ chính xác) để đưa vào model và nhận diện. Model được sử dụng trong ứng dụng này đang là model PCA thuần.

Hình 8: Giao diện của ứng dụng nhận diện khuôn mặt

10 Thảo luận

10.1 Ưu và nhược điểm của PCA

Ưu điểm

PCA đã được ứng dụng nhiều trong việc nhận diện khuôn mặt nhờ vào các điểm mạnh như:

- **Khả năng giảm chiều dữ liệu:** Các hình ảnh là những ma trận với kích thước rất lớn, gây khó khăn cho việc xử lý và lưu trữ. PCA giúp biểu diễn dữ liệu trong không gian có chiều thấp hơn nhưng vẫn giữ lại phần lớn thông tin quan trọng.
- **Trực quan hóa tốt:** PCA cho phép biểu diễn dữ liệu dưới dạng các *eigenfaces* (các thành phần chính), giúp ta trực quan hóa các đặc trưng tổng quát của khuôn mặt một cách rõ ràng và có ý nghĩa.
- **Hiệu quả về mặt tính toán:** Việc giảm số chiều dữ liệu đầu vào giúp tăng tốc độ xử lý, giảm chi phí lưu trữ và thời gian huấn luyện các mô hình nhận diện khuôn mặt.
- **Không cần nhiều dữ liệu huấn luyện:** PCA không yêu cầu một lượng lớn mẫu dữ liệu để hoạt động hiệu quả, đặc biệt phù hợp trong bối cảnh dữ liệu có giới hạn.
- **Khử nhiễu:** PCA có khả năng loại bỏ nhiễu trong dữ liệu bằng cách giữ lại các thành phần chính có phương sai lớn và loại bỏ các thành phần không đáng kể.

Nhược điểm

Mặc dù PCA là một công cụ mạnh mẽ, nhưng nó vẫn tồn tại nhiều hạn chế:

- **Nhạy cảm với điều kiện ánh sáng:** Sự thay đổi ánh sáng có thể ảnh hưởng đáng kể đến kết quả trích chọn đặc trưng và độ chính xác của việc nhận diện.
- **Thay đổi tư thế khuôn mặt:** Khi người dùng nghiêng đầu, quay mặt hoặc thay đổi góc nhìn, hệ thống có thể gặp khó khăn trong việc nhận dạng chính xác.
- **Biểu cảm khuôn mặt:** Các cảm xúc như cười, nheo mắt, hay há miệng cũng làm thay đổi đặc trưng của khuôn mặt, gây khó khăn cho việc nhận diện.
- **Biến đổi góc nhìn:** Khi khuôn mặt bị xoay ở các góc khác nhau, độ chính xác thường giảm đáng kể.
- **Nhận diện từ video:** Mặc dù dữ liệu video chứa nhiều thông tin phong phú, nhưng phần lớn hệ thống hiện nay chỉ xử lý các khung ảnh tĩnh, chưa tận dụng được lợi ích của dòng video liên tục.
- **Lão hóa khuôn mặt:** Khuôn mặt con người thay đổi theo thời gian, khiến hệ thống cần liên tục cập nhật để đảm bảo độ chính xác.
- **Độ phân giải thấp:** Hình ảnh từ camera giám sát thường có chất lượng thấp, gây khó khăn cho việc nhận dạng chính xác.
- **Quy mô hệ thống:** Trong thực tế, cơ sở dữ liệu khuôn mặt có thể chứa hàng triệu đến hàng tỷ ảnh, đặt ra bài toán về hiệu năng và tốc độ xử lý.

Ứng dụng thực tế của PCA trong nhận diện khuôn mặt

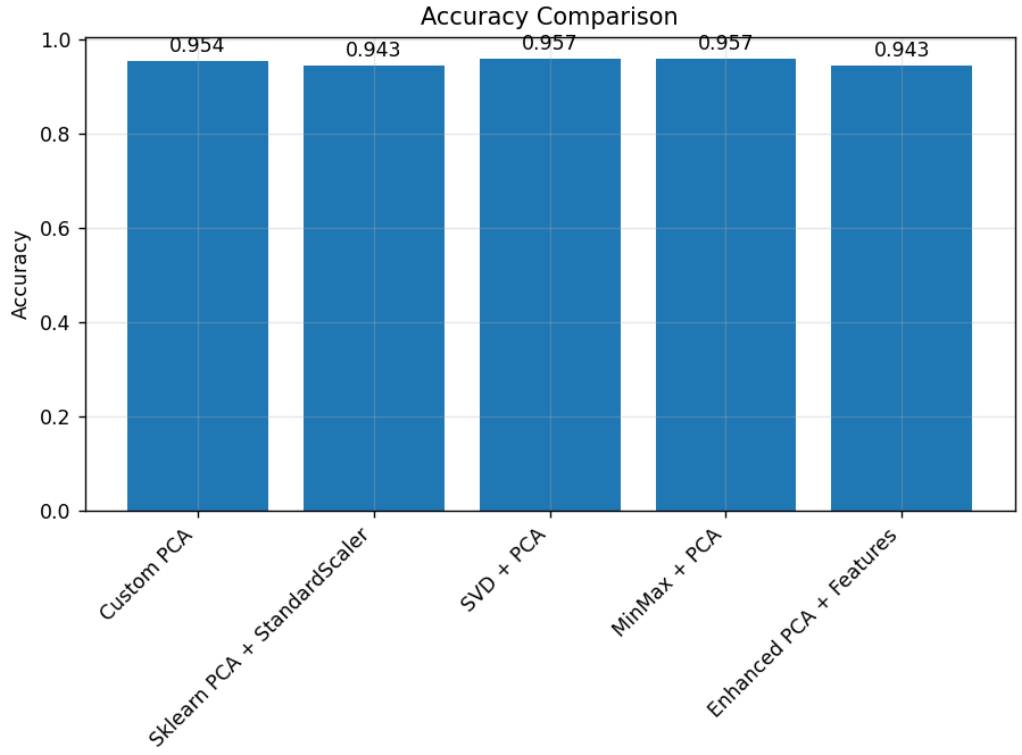
- **An ninh và pháp luật:** Hỗ trợ nhận dạng tội phạm, truy vết đối tượng trong các hệ thống camera giám sát.
- **Xác minh danh tính:** Sử dụng trong thẻ căn cước, hộ chiếu, hệ thống kiểm tra chứng minh thư.
- **Kiểm soát truy cập:** Dùng để mở khoá thiết bị hoặc truy cập vào hệ thống làm việc.
- **Giao dịch tài chính:** Tăng cường bảo mật trong các giao dịch ngân hàng, rút tiền từ ATM.
- **Tương tác người – máy:** Hỗ trợ người khuyết tật giao tiếp thông qua biểu cảm khuôn mặt.
- **Giám sát trạng thái lái xe:** Phát hiện tài xế buồn ngủ, mất tập trung và cảnh báo kịp thời.
- **Tìm kiếm hình ảnh:** Hỗ trợ tra cứu dữ liệu khuôn mặt trên mạng xã hội, truyền hình, hoặc các hệ thống lưu trữ lớn như internet.
- **Điện thoại và thiết bị di động:** Tự động phân loại ảnh trong album theo từng người hoặc dùng để mở khóa thiết bị.
- **Điều khiển robot:** Robot có thể nhận diện người dùng và tương tác phù hợp hơn.

10.2 Cải tiến PCA và phân tích kết quả

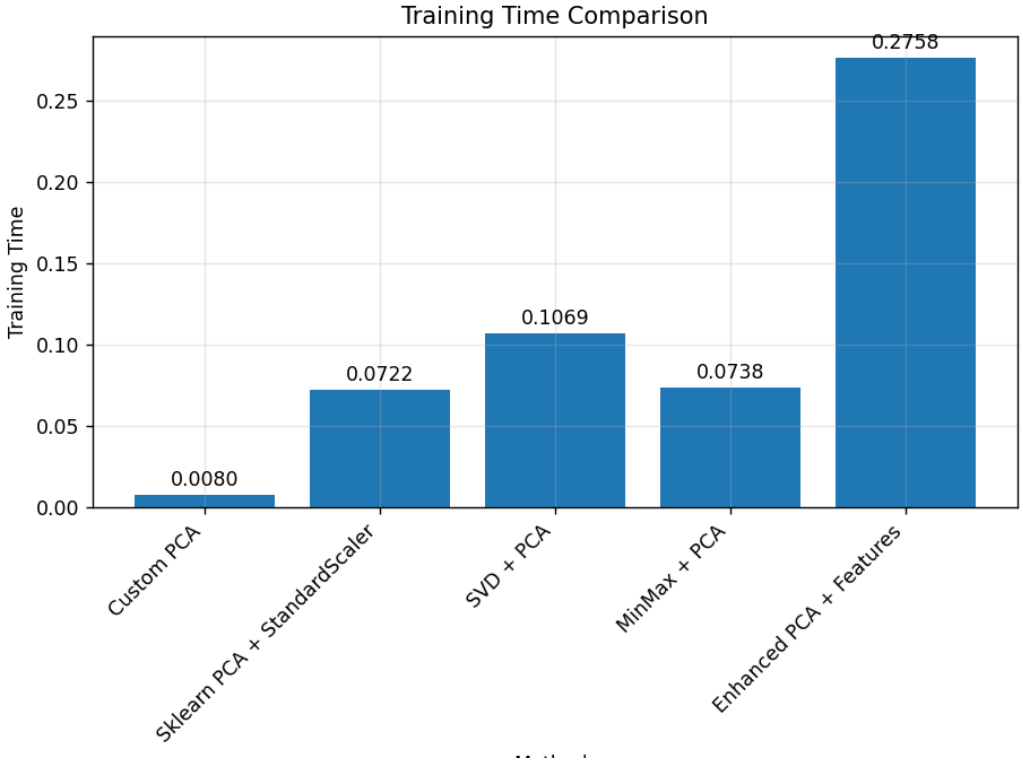
Nhóm đã sử dụng 4 phương pháp khác nhau để so sánh kết quả với PCA

- **Sử dụng hàm PCA có sẵn trong thư viện scikit-learn cùng với standard scaler**
- **Sử dụng SVD trước khi đưa vào PCA**
- **Sử dụng hàm PCA có sẵn trong thư viện scikit-learn cùng với MinMax scaler**
- **Sử dụng PCA trên các dữ liệu được xử lý sâu(feature engineered):** Do PCA nhạy cảm với các giá trị outlier, trong phương pháp này, nhóm thêm các giá trị thống kê của hình ảnh vào ma trận ảnh trước khi đưa qua PCA

Sau đây là các kết quả của cả 5 phương pháp khác nhau



Hình 9: Accuracy giữa các phương pháp



Hình 10: Thời gian huấn luyện giữa các phương pháp

Với độ chính xác không quá khác biệt giữa các phương pháp nhưng đổi lại thời gian tính toán gấp nhiều lần so với PCA thuần. SVD +PCA và MinMax + PCA dẫn đầu

trong độ chính xác, thời gian tính toán của cả hai phương pháp đều ở mức ổn. Với việc thêm các giá trị thống kê của hình ảnh vào ma trận ảnh, chi phí tính toán của phương pháp cuối trở nên khá đắt. Việc chọn giữa độ chính xác và độ phức tạp trong tính toán sẽ tùy vào các bài toán mà ta chọn các phương pháp tương ứng.

11 Kết luận

PCA là một trong những thuật toán giảm chiều có tính ứng dụng cao trong việc nhận diện khuôn mặt nhờ vào khả năng giảm lượng dữ liệu mà vẫn giữ lại được phần lớn các đặc trưng của dữ liệu. Do đó các khuôn mặt có thể được nhận diện một cách dễ dàng. Tuy nhiên, PCA là một thuật toán khá đơn giản nên vẫn còn gặp nhiều vấn đề khi sử dụng trên các môi trường ánh sáng, biểu cảm, tư thể khác nhau của khuôn mặt. SVD trước khi PCA hoặc một số thuật toán khác có thể cải thiện được nhược điểm đó nhưng đổi lại chi phí tính toán sẽ cao hơn PCA thuần. Với các hệ thống nhận diện khuôn mặt real-time trên các phần cứng hạn chế thì PCA chính là ứng cử viên hoàn hảo nhờ vào các ưu điểm trên.

12 Tài liệu tham khảo

Tài liệu

[1] Vũ Hữu Tiệp. *Principal Component Analysis*. Machine Learning Cơ Bản. https://machinelearningcoban.com/2017/06/15/pca/

[2] Viblo. *Thuật toán giảm chiều dữ liệu PCA*. https://viblo.asia/p/ml-from-scratch-thuat-toan-giam-chieu-du-lieu-pca-7ymJXKMa4kq

[3] Studocu. *Tài liệu PCA*. https://www.studocu.vn/vn/document/81858188

[4] M. Turk and A. Pentland. *Eigenfaces for Recognition*. In: *Cognitive Neuroscience: Trends in Cognitive Sciences*. Springer, Boston, MA, 1998. DOI: https://doi.org/10.1007/0-387-22440-8_13

[5] YoshioTakane. *Constrained Principal Component Analysis and Related Techniques*. Chapman & Hall/CRC, Boca Raton, 2013. ISBN: 978-1466556669. https://www.taylorfrancis.com/books/mono/10.1201/b16020/constrained-principal-component-analysis-related-techniques-yoshio-takane