TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN

TOÁN ỨNG DỤNG THỐNG KÊ

BÁO CÁO THỰC HÀNH

Project Cuối Kì: Phương Pháp PCA Để Giảm Chiều Dữ Liệu

Mã số sinh viên: 21120542 - 21120582

Họ và Tên: Lâm Hoàng Quốc – Đinh Hoàng Trung

Mục Lục

I. PHÂN CÔNG:

A. THÔNG TIN NHÓM:

- Sinh viên 1: 21120542 – Lâm Hoàng Quốc.
- Sinh viên 2: 21120582 – Đinh Hoàng Trung.
- Chủ đề chọn: 1 (Đại số tuyến tính):

"Phương pháp Principal Component Analysis (PCA) để giảm chiều dữ liệu"

B. PHÂN CÔNG CỤ THỂ:

CÔNG VIỆC	NGƯỜI THỰC HIỆN
Viết Báo Cáo – Làm Bìa	Trung
Viết Báo Cáo – Làm Mục Lục	Trung
Viết Báo Cáo – Viết Bố Cục	Quốc
Viết Báo Cáo – Thông Tin Nhóm & Bảng Phân Công	Quốc
Viết Báo Cáo – Giải Thích PCA, Liên Hệ SVD	Trung
Viết Báo Cáo – Kết Luận	Quốc
Viết Báo Cáo – Tài Liệu Tham Khảo	Trung,Quốc
Tìm Hiểu, Code Và Quay Video Demo Ứng Dụng 1	Trung
Tìm Hiểu, Code Và Quay Video Demo Ứng Dụng 2	Quốc

II. GIẢI THÍCH PCA, LIÊN HỆ SVD:

A. KIẾN THỨC CHUẨN BỊ:

- **Norm 2 của ma trận:**
- **Biểu diễn vector trong các hệ cơ sở khác nhau:**
- **Trace:** Tổng tất cả giá trị trên đường chéo chính của ma trận vuông.
- **Kì vọng và ma trận hiệp phương sai:**
  - Trong dữ liệu 1 chiều: Kì vọng $\bar{x}$ và Phương sai σ^2 lần lượt là.

$$\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x_n = \frac{1}{N}\mathbf{X1}$$

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

  - **Trong dữ liệu nhiều chiều:** Kì vọng $\bar{x}$ và Phương sai S lần lượt là.

    Cho N điểm dữ liệu được biểu diễn bởi các cột x_1, ……, x_N, khi đó:

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T = \frac{1}{N}\hat{\mathbf{X}}\hat{\mathbf{X}}^T$$

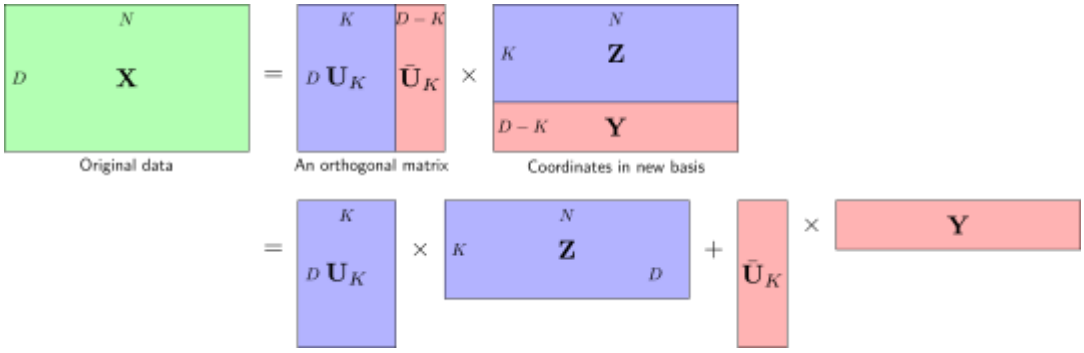
Khá tương đồng với các công thức cho dữ liệu một chiều. Có một vài điều cần lưu ý.

- Ma trận hiệp phương sai là ma trận đối xứng và là ma trận nửa xác định dương.
- Mọi phần tử trên đường chéo chính không âm. Chúng chính là phương sai của từng chiều dữ liệu.
- Các phần tử S_{ij} ($i \neq j$) thể hiện sự tương quan giữa phần tử thứ I và thứ j của dữ liệu. Giá trị này có thể dương, âm hoặc bằng 0. Khi bằng 0, ta nói rằng thành phần I và j không tương quan.
- Nếu ma trận là ma trận dường chéo chứng tỏ dữ liệu hoàn toàn không tương quan giữa các chiều.

- **Đặt vấn đề:**
  - Để giảm chiều dữ liệu từ D về K với K < D là giữ lại K phần tử quan trọng nhất.
  - Tuy nhiên, việc làm này chắc chắn chưa phải tốt nhất, do ta chưa biết xác định phần nào quan trọng hơn. Hoặc trong một số trường hợp, lượng thông tin mà mỗi thành phần mang đều như nhau, bỏ đi thành phần nào cũng dẫn đến việc mất một lượng thông tin lớn.
  - Nếu chúng ta biểu diễn Vector ban đầu trong một hệ cơ sở mới mà trong hệ cơ sở đó mà tầm quan trọng giữa các thành phần khác nhau rõ rệt, thì chúng ta có thể bỏ qua những thành phần ít quan trọng nhất.
    - Ví dụ: Các góc nhìn từ bên ngoài nhìn vào một tòa nhà nhiều tầng, thì so với góc nhìn từ phía chính diện từ xa mang nhiều thông tin hơn so với góc nhìn từ trên đỉnh tòa nhà nhìn xuống. Vì vậy, từ góc nhìn phía trên đỉnh tòa nhà ta có thể được bỏ qua mà không có quá nhiều thông tin về hình dáng tòa nhà bị mất.

B. GIẢI THÍCH PCA:

- **Định Nghĩa:** PCA là phương pháp đi tìm một hệ cơ sở mới sao cho thông tin của dữ liệu chủ yếu chỉ tập trung ở một vài tọa độ, phần còn lại chỉ mang một lượng nhỏ thông tin.
- **Lý thuyết:** Tìm một hệ trực chuẩn mới sao cho trong hệ này, các thành phần quan trọng nhất nằm trong K thành phần đầu tiên.



- Mục đích của PCA là tìm ma trận trực giao U sao cho phần thông tin lớn nhất được giữ lại ở phần xanh $\mathbf{U_kZ}$ và phần đỏ $\bar{\boldsymbol{U}}_{\mathbf{K}}\mathbf{Y}$ được lược bỏ và thay bằng một ma trận không phụ thuộc vào từng điểm dữ liệu.

$$\mathbf{X} = \mathbf{U}_K\mathbf{Z} + \bar{\mathbf{U}}_K\mathbf{Y} \quad \textbf{(1)}$$

Từ đó suy ra:

$$\begin{bmatrix}\mathbf{Z}\\ \mathbf{Y}\end{bmatrix} = \begin{bmatrix}\mathbf{U}_K^T\\ \bar{\mathbf{U}}_K^T\end{bmatrix}\mathbf{X} \Rightarrow \begin{matrix}\mathbf{Z} = \mathbf{U}_K^T\mathbf{X}\\ \mathbf{Y} = \bar{\mathbf{U}}_K^T\mathbf{X}\end{matrix} \quad \textbf{(2)}$$

- Ta sẽ xấp xỉ Y với một ma trận có toàn bộ cột như sau (gọi mỗi cột đó là b):

$$\mathbf{Y} \approx \mathbf{b}.\mathbf{1}^T$$

  - Với $\mathbf{1}^T \in \mathbf{R}^{1xN}$ là vector có toàn bộ phần tử bằng 1. Giả sử đã tìm được U, ta cần tìm **b** thỏa mãn:

$$\mathbf{b} = \mathbf{argmin_b}||\mathbf{Y} - \mathbf{b}\mathbf{1}^T||_F^2 = \mathbf{argmin_b}||\bar{\mathbf{U}}_K^T\mathbf{X} - \mathbf{b}\mathbf{1}^T||_F^2$$

  - Giải phương trình đạo hàm theo b bằng 0 ta được:

$$(\mathbf{b}\mathbf{1}^T - \bar{\mathbf{U}}_K^T\mathbf{X})\mathbf{1} = 0 \Rightarrow N\mathbf{b} = \bar{\mathbf{U}}_K^T\mathbf{X}\mathbf{1} \Rightarrow \mathbf{b} = \bar{\mathbf{U}}_K^T\bar{\mathbf{x}}$$

  - Như vậy, việc tính toán sẽ thuận tiện hơn nếu **vector kì vọng $\bar{\boldsymbol{x}}$ = 0**. Vì vậy bước đầu tiên của PCA là trừ mỗi vector dữ liệu đi kì vọng của toàn bộ dữ liệu.

- Với giá trị b tìm được, dữ liệu ban đầu sẽ xấp xỉ với:

$$\mathbf{X} \approx \tilde{\mathbf{X}} = \mathbf{U}_K\mathbf{Z} + \bar{\mathbf{U}}_K\bar{\mathbf{U}}_K^T\bar{\mathbf{x}}\mathbf{1}^T \quad \textbf{(3)}$$

- Từ **(1), (2), (3)** ta định nghĩa **hàm mất mát** chính:

$$J = \frac{1}{N}||\mathbf{X} - \tilde{\mathbf{X}}||_F^2 = \frac{1}{N}||\bar{\mathbf{U}}_K\bar{\mathbf{U}}_K^T\mathbf{X} - \bar{\mathbf{U}}_K\bar{\mathbf{U}}_K^T\bar{\mathbf{x}}\mathbf{1}^T||_F^2 \quad \textbf{(4)}$$

- Mà nếu các cột của một ma trận V tạo thành một hệ trực chuẩn thì với một ma trận W bất kỳ, ta luôn có:

$$||\mathbf{VW}||_F^2 = \text{trace}(\mathbf{W}^T\mathbf{V}^T\mathbf{VW}) = \text{trace}(\mathbf{W}^T\mathbf{W}) = ||\mathbf{W}||_F^2$$

- Áp dụng định lý trên vào **(4)**, hàm mất mát có thể được viết lại thành:

$$\begin{aligned} J &= \frac{1}{N}||\bar{\mathbf{U}}_K^T(\mathbf{X} - \bar{\mathbf{x}}\mathbf{1})^T||_F^2 = \frac{1}{N}||\bar{\mathbf{U}}_K^T\hat{\mathbf{X}}||_F^2 \\ &= \frac{1}{N}||\hat{\mathbf{X}}^T\bar{\mathbf{U}}_K||_F^2 = \frac{1}{N}\sum_{i=K+1}^{D}||\hat{\mathbf{X}}^T\mathbf{u}_i||_2^2 \\ &= \frac{1}{N}\sum_{i=K+1}^{D}\mathbf{u}_i^T\hat{\mathbf{X}}\hat{\mathbf{X}}^T\mathbf{u}_i \\ &= \sum_{i=K+1}^{D}\mathbf{u}_i^T\mathbf{S}\mathbf{u}_i \end{aligned} \quad \textbf{(5)}$$

  - Với $\hat{\boldsymbol{X}} = \mathbf{X} - \bar{\boldsymbol{x}}.\mathbf{1}^T$ là dữ liệu đã chuẩn hóa
  - **S** là ma trận hiệp phương sai của dữ liệu.
- Việc cần làm còn lại là tìm $\mathbf{u}_i$ sao cho mất mát nhỏ nhất. Thay **K = 0** vào **(5)**:

$$\begin{aligned} L &= \sum_{i=1}^{D}\mathbf{u}_i^T\mathbf{S}\mathbf{u}_i = \frac{1}{N}||\hat{\mathbf{X}}^T\mathbf{U}||_F^2 \\ &= \frac{1}{N}\text{trace}(\hat{\mathbf{X}}^T\mathbf{U}\mathbf{U}^T\hat{\mathbf{X}}) \\ &= \frac{1}{N}\text{trace}(\hat{\mathbf{X}}^T\hat{\mathbf{X}}) \\ &= \frac{1}{N}\text{trace}(\hat{\mathbf{X}}\hat{\mathbf{X}}^T) \\ &= \text{trace}(\mathbf{S}) = \sum_{i=1}^{D}\lambda_i \end{aligned}$$

  - Với $\boldsymbol{\lambda}_i$ là các giá trị riêng của ma trận nửa xác định dương **S**.
- Suy ra **L** chính là tổng của các phương sai theo từng thành phần của dữ liệu ban đầu.
- Vì vậy, việc **làm tối thiểu mất mát J** được cho bởi **(5)**, là **làm tối đa L – J**:

$$F = L - J = \sum_{i=1}^{K}\mathbf{u}_i\mathbf{S}\mathbf{u}_i^T$$

- **Định Lý:**
  - **F** đặt giá trị lớn nhất bằng $\sum_{i=1}^{K}\boldsymbol{\lambda}_i$ khi $\mathbf{u}_i$ là các vector riêng có norm 2 (khoảng cách Euclid) bằng 1 ứng với các giá trị riêng này (các u_i phải trực giao với nhau).
  - $\boldsymbol{\lambda}_i$ **(i = 1, 2, …, K)** chính là **K** giá trị riêng lớn nhất của a trận hiệp phương sai **S**.

- Ta chỉ giữ K thành phần chính của giữ liệu khi muốn giảm số chiều dữ liệu.
- **Vì vậy:** PCA còn được coi là phương pháp giảm số chiều dữ liệu mà giữ được tổng phương sai còn lại là lớn nhất.

- **Các bước thực hiện PCA:**

1. Tính vector kì vọng của toàn bộ dữ liệu:

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

2. Trừ mỗi điểm dữ liệu đi vector kì vọng của toàn bộ dữ liệu:

$$\hat{\mathbf{x}}_n = \mathbf{x}_n - \bar{\mathbf{x}}$$

3. Tính ma trận hiệp phương sai:

$$\mathbf{S} = \frac{1}{N}\hat{\mathbf{X}}\hat{\mathbf{X}}^T$$

4. Tính các trị riêng và vector riêng có norm bằng 1 của ma trận này, sắp xếp chúng theo thứ tự giảm dần của trị riêng.
5. Chọn K vector riêng ứng với K trị riêng lớn nhất để xây dựng ma trận U_K có các cột tạo thành một hệ trực giao. K vector này, còn được gọi là các thành phần chính, tạo thành một không gian con gần với phân bố dữ liệu ban đầu đã chuẩn hóa.
6. Chiếu dữ liệu ban đầu đã chuẩn hóa $\hat{X}$ xuống không gian con tìm được.
7. Dữ liệu mới chính là tọa đồi của các điểm dữ liệu trên không gian mới:

$$\mathbf{Z} = \mathbf{U}_K^T\hat{\mathbf{X}}$$

⇨ **Dữ liệu ban đầu được xấp xỉ theo dữ liệu mới như sau:**

$$\mathbf{x} \approx \mathbf{U}_K\mathbf{Z} + \bar{\mathbf{x}}$$

C. MỐI LIÊN HỆ GIỮA SVD vs PCA:

- **Nhắc lại SVD:**
  - SVD của $X \in R^{DxN}$ là:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \quad \textbf{(1)}$$

    - Với $U \in \mathbb{R}^{DxD}$ và $V \in \mathbb{R}^{NxK}$ là các ma trận trực giao
    - $\Sigma \in R^{DxN}$ là ma trận đường chéo (có thể không vuông) với các phần tử trên đường chéo không âm giảm dần.
  - Bài toán xấp xỉ low-rank tốt nhất:

$$\min_{\mathbf{A}} ||\mathbf{X} - \mathbf{A}||_F$$
$$\text{s.t. } \text{rank}(\mathbf{A}) = K \quad \textbf{(2)}$$

- Với SVD của X đã nêu trên: Nghiệm của bài toán sẽ là:

$$\mathbf{A} = \mathbf{U}_K \mathbf{\Sigma}_K \mathbf{V}_K^T \quad \textbf{(3)}$$

- Với U ∈ $\mathbb{R}^{DxK}$ và V V ∈ $\mathbb{R}^{NxK}$ là các ma trận tạo bởi K cột đầu tiên của U và V, và $\Sigma_K \in \mathbb{R}^{KxK}$ là ma trận đường chéo con tương ứng với K hàng đầu tiên và K cột đầu tiên của Σ.

- **Ý Tưởng PCA:**
  - PCA là bài toàn tìm ma trận trực giao U và ma trận mô tả dữ liệu không gian thấp chiều Z sao cho xấp xỉ sau tốt nhất:

$$\mathbf{X} \approx \tilde{\mathbf{X}} = \mathbf{U}_K \mathbf{Z} + \bar{\mathbf{U}}_K \bar{\mathbf{U}}_K^T \bar{\mathbf{x}} \mathbf{1}^T$$

  - Với $\mathbf{U_K}$, $\bar{\boldsymbol{U}}_\mathbf{K}$ lần lượt là các ma trận được tạo bởi **K** cột đầu tiên và **D - K** cuối cuối cùng của **U**, và $\bar{\boldsymbol{x}}$ là vector kỳ vọng của dữ liệu.
  - Như đã nêu trên lý thuyết, giả sửa $\bar{\boldsymbol{x}} = \mathbf{0}$, khi đó:

$$\mathbf{X} \approx \tilde{\mathbf{X}} = \mathbf{U}_K \mathbf{Z}$$

  - Bài toán tối ưu sẽ thành:

$$\mathbf{U}_K, \mathbf{Z} = \min_{\mathbf{U}_K, \mathbf{Z}} ||\mathbf{X} - \mathbf{U}_K \mathbf{Z}||_F$$
$$\text{s.t.:} \quad \mathbf{U}_K^T \mathbf{U}_K = \mathbf{I}_K \quad \textbf{(4)}$$

    - Với $\mathbf{I_K} \in \mathbb{R}^{\mathbf{KxK}}$ là ma trận tạo đơn vị trong không gian **K** chiều, và điều kiện ràng buộc là tạo thành hệ trực chuẩn.

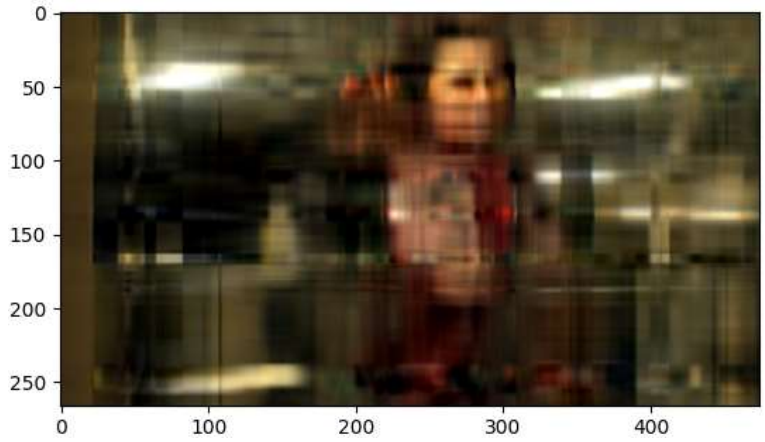
- **Quan hệ giữa SVD và PCA:**
  - Ta có thể thấy được sự tương đồng của 2 bài toán tối ưa **(2)** và **(4)**, với nghiệm của bài toán **(2)** được cho trong **(3)**.
  - Suy ra, nghiệm của bài toán **(4)**:
    - $\mathbf{U_K}$ **trong (4) =** $\mathbf{U_K}$ **trong (3)**
    - **Z trong (4) =** $\mathbf{\Sigma_K V^T_K}$ **trong (3)**
  - **Như vậy:** Nếu các điểm dữ liệu được biểu diễn bởi các cột của một ma trận và trung bình của mỗi hàng của ma trận đó bằng 0, thì nghiệm của bài toán PCA chính là trường hợp đặc biệt của bài toán Matrix Factorization giải bằng SVD.

III. HAI ỨNG DỤNG VÀ DEMO:

A. ỨNG DỤNG 1: COMPRESS PICTURE:

- **Công Dụng:** giảm thiểu dung lượng của tệp hình ảnh đồ họa, mà không làm giảm chất lượng của hình ảnh trong phạm vi có thể chấp nhận được.
- **Thuật Toán:** Sử dụng PCA để tìm số thành phần chính.
    - Chọn số lượng thành phần chính.
    - Bằng cách loại bỏ các thành phần ít quan trọng là giữ lại các thành phần chính ta có thể giảm thiểu dung lượng ảnh một cách đáng kể mà không giảm chất lượng ảnh trong phạm vi chấp nhận được.
- **Ghi chú:**
    - Số thành phần chính chọn ra trong bài lần lượt là 8, 32, 64.
    - Ảnh input được đọc cứng trong code.
- **Định dạng input và output:**
    - **Input:** ảnh được nhập cứng với tên ảnh là ‘input_compress.png’.
    - **Output:** ảnh output lần lượt là ‘ouput_compress’ + <số thành phần chính được lấy khi compress> + ‘.png’.
        - **output_compress8.png**
        - **output_compress32.png**
        - **output_compress64.png**
- **Demo:**
    - Input:

    - Output:
        - **output_compress8.png**



Với 8 thành phần ta có thể thể hiện 83.46162121742964% phương sai

- **output_compress32.png**

Với 32 thành phần ta có thể thể hiện 96.55130858300254% phương sai

- **output_compress64.png**

Với 64 thành phần ta có thể thể hiện 98.90402773744427% phương sai

B. ỨNG DỤNG 2: DENOISE PICTURE:

- **Công Dụng:** Từ một bức ảnh bị nhiễu, giảm nhiễu và lưu lại thành một ảnh khác nét hơn.
- **Thuật Toán:** Sử dụng thuật toán PCA làm trọng tâm:
  - Chọn ra số lượng thành phần chính, (cụ thể em chọn 50).
  - Các thành phần nhiễu thường có biến thiên cao. Sử dụng PCA cho phép tách biệt giữa thành phần nhiễu và thành phần chính trong ảnh
  - Bằng cách giữ lại các thành phần chính quan trọng nhất và loại bỏ thành phần nhiễu, ta có thể giảm kích thước của ảnh và loại bỏ nhiễu. Từ đó tái tạo lại ảnh ít nhiễu gần giống ảnh gốc.
- **Ghi chú:**
  - Khi ảnh được giảm nhiễu bằng thuật toán PCA, ảnh sẽ chuyển về màu xám vì các thành phần chính được sắp xếp theo độ quan trọng của chúng và không có màu sắc
  - Vì em không có sẵn những bức ảnh nhiễu, và cho linh hoạt thì em có viết hàm để tạo một ảnh nhiễu từ ảnh đọc đầu vào. Độ nhiễu em chọn trong code là 0,05.
  - Trong source code, có một vài dòng code em đã comment khóa chúng lại. Những dòng code đó sẽ giúp hiện những bức ảnh có hiện diện trong chương trình lên màn hình để thầy có thể xem ngay lập tức luôn mà không cần phải mở những tấm hình được lưu sau khi kết thúc chương trình. Nếu muốn thì thầy hãy mở khóa chúng ra để sử dụng luôn nhé ạ.
- **Định dạng input và output:**
  - Input: một dòng nhập vào, là tên của file ảnh (định dạng .jpg; .png; …)
  - Output: Chương trình chạy xong sẽ lưu ra hai bức ảnh:
    - "noise_picture.jpg": là ảnh nhiễu tạo ra từ "input_picture.jpg".
    - "pca_picture.jpg": là ảnh kết quả từ việc khử nhiễu sử dụng phương pháp PCA lên ảnh "noise_picture.jpg"
- **Demo:**

Input	input_picture1.jpg	input_picture2.png
Output		

noise_picture.jpg		
pca_picture.jpg		

IV. KẾT LUẬN:

Tổng quan, PCA là một công cụ hữu ích trong việc nén ảnh và giảm nhiễu ảnh. Nó cho phép giảm kích thước dữ liệu, không gian lưu trữ và cải thiện chất lượng ảnh bằng cách xác định các thành phần chính quan trọng và loại bỏ thành phần không quan trọng. Tuy nhiên, việc sử dụng PCA cần được điều chỉnh và cân nhắc để đảm bảo cân đối giữa kích thước và chất lượng của ảnh sau khi xử lý.

V. NGUỒN THAM KHẢO:

THAM KHẢO TRÊN MẠNG:

- Ứng dụng PCA để khử nhiễu ảnh (PDF) PCA based image denoising (researchgate.net)
- Áp dụng khử nhiễu ảnh sử dụng ngôn ngữ Python: (195) Python#021 Image Denoising using Principal Component Analysis (PCA) in Python - YouTube
- Định Nghĩa Norm của ma trận: Machine Learning cơ bản (machinelearningcoban.com)
- Giải thích PCA: Machine Learning cơ bản (machinelearningcoban.com)
- Mỗi liên hệ của PCA và SVD: Machine Learning cơ bản (machinelearningcoban.com)
- Demo PCA: erdogant/pca: pca: A Python Package for Principal Component Analysis. (github.com)
- Nguồn ảnh Compress: Ảnh cá nhân và phim “Everything Everywhere All At Once”.