

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & ỨNG DỤNG



BÀI TẬP LỚN ĐẠI SỐ TUYẾN TÍNH (MT1007)

Đề tài 4

PHÂN TÍCH SVD (SINGULAR VALUE DECOMPOSITION)
ĐỂ NÉN DỮ LIỆU

LỚP DT03 – NHÓM 3 – HK243
NGÀY NỘP 08/08/2025

Giảng viên hướng dẫn: ThS.Nguyễn Hữu Hiệp

Sinh viên thực hiện	Mã số sinh viên
Nguyễn Duy Khang	2533022
Phạm Công Võ	2313946
Phan Kế Vĩnh Hưng	2111412
Trần Khánh An	2310037
Lê Như Ý	2414094
Trần Vũ Hữu Phúc	2014187
Võ Nguyễn Tường Vi	2313882

Tp. Hồ Chí Minh, Tháng 8/2025

PHÂN CÔNG NHIỆM VỤ VÀ KẾT QUẢ THỰC HIỆN ĐỀ TÀI CỦA TỪNG THÀNH VIÊN NHÓM 4

STT	Họ và tên	MSSV	Nhiệm vụ	Kết quả	Chữ ký
1	Nguyễn Duy Khang	2533022	Thiết kế PowerPoint	100%	
2	Phạm Công Võ	2313946	Soạn Thảo Nội Dung + Phân Tích SVD	100%	
3	Phan Kế Vĩnh Hưng	2111412	Xây dựng phần mềm (Code Python) giảm chiều của một file dữ liệu	100%	
4	Trần Khánh An	2310037	Thuyết Trình	100%	
5	Lê Như Ý	2414094	Thuyết Trình	100%	
6	Trần Vũ Hữu Phúc	2014187	Cơ Sở Lí Thuyết	100%	
7	Võ Nguyễn Tường Vi	2318882	Ứng Dụng SVD Trong Giảm Chiều Dữ Liệu	100%	

Mục lục

1	Lời Mở Đầu	4
2	Tổng Quan Về Nén Dữ Liệu Bằng Phương Pháp SVD	5
2.1	Giới thiệu	5
2.2	Lịch sử phát triển và Vai trò	5
2.2.1	Lịch sử hình thành và phát triển	5
2.2.2	Vai trò trong thời đại dữ liệu số	5
2.3	Phương pháp và ứng dụng nén dữ liệu sử dụng SVD	5
2.3.1	Phương pháp nén dữ liệu bằng SVD	5
2.3.2	Ứng dụng thực tiễn	6
3	Cơ Sở Lí Thuyết	6
3.1	Trị riêng và vector riêng	6
3.2	Chéo hóa ma trận	7
3.3	Hệ trục giao và trục chuẩn	7
4	Phân Tích SVD (Singular Value Decomposition)	8
4.1	Singular Value Decomposition (SVD)	8
4.1.1	Biểu diễn tổng quát	8
4.1.2	Mối liên hệ giữa SVD và phân tích trị riêng	9
4.1.3	Tính chất của phân tích SVD	10
4.2	Compact SVD (Rút Gọn SVD)	10
4.3	Truncated SVD (Phép Xấp Xỉ SVD)	11
5	Ứng Dụng SVD Để Nén Dữ Liệu	12
5.1	Phương thức lưu trữ dữ liệu hình ảnh trong máy tính	12
5.2	Xử lý dữ liệu hình ảnh bằng SVD	13
5.3	Tính toán dung lượng lưu trữ và tỷ lệ nén ảnh với Truncated SVD	14
6	Code Python	15
6.1	Giới thiệu Python	15
6.2	Các hàm thường dùng trong Python	15
6.3	Code python	16
6.4	Kết quả chạy code	17
	Tài Liệu Tham Khảo	20

Danh sách hình vẽ

1	Minh họa phân rã SVD cho ma trận A trong hai trường hợp: $m < n$ (trên) và $m > n$ (dưới). Ma trận Σ là đường chéo, các giá trị giảm dần và không âm; màu đỏ đậm biểu thị giá trị lớn, ô trắng biểu thị giá trị bằng 0.	9
2	Biểu diễn SVD dạng thu gọn và biểu diễn ma trận dưới dạng tổng các ma trận có rank bằng 1.	11
3	Minh họa ma trận điểm ảnh	12
4	Mô hình ba ma trận RGB	12
5	Minh họa quá trình loại bỏ các trị riêng trên đường chéo	14
6	So sánh ảnh gốc và ảnh tái tổ hợp theo k	18
7	Giá trị của các singular values của ma trận ảnh theo k	18

1 Lời Mở Đầu

Trong bối cảnh thế giới đang bước vào kỷ nguyên của dữ liệu và tự động hóa, việc xử lý và khai thác thông tin một cách thông minh không chỉ là nhu cầu mà đã trở thành yêu cầu tất yếu trong hầu hết các lĩnh vực khoa học và công nghệ. Giữa làn sóng dữ liệu khổng lồ và liên tục phát sinh, việc nén dữ liệu – làm cho thông tin trở nên tinh gọn hơn nhưng vẫn giữ được giá trị cốt lõi – là một bài toán không thể thiếu trong mọi hệ thống hiện đại.

Trong dòng chảy đó, phân tích giá trị suy biến (Singular Value Decomposition – SVD) đã nổi lên như một công cụ toán học mạnh mẽ, kết tinh từ nền tảng sâu sắc của đại số tuyến tính. SVD không chỉ là kỹ thuật phân rã ma trận thuần túy, mà còn là biểu tượng của cách mà tư duy toán học có thể giải quyết những vấn đề thực tiễn phức tạp: từ giảm chiều dữ liệu, lọc nhiễu, đến trích xuất đặc trưng phục vụ học máy và trí tuệ nhân tạo. Trong từng phép biến đổi và kết cấu toán học ấy là một thế giới của sự tinh giản, tối ưu và đầy ý nghĩa.

Đề tài "Ứng dụng phân tích SVD để nén dữ liệu" chính là hành trình mà nhóm chúng em lựa chọn để khám phá chiều sâu của toán học trong một ứng dụng công nghệ rất cụ thể. Qua việc tiếp cận và triển khai đề tài này, chúng em không chỉ nâng cao kiến thức chuyên môn mà còn được rèn luyện tư duy phân tích, kỹ năng mô hình hóa và khả năng gắn kết lý thuyết với thực tiễn – những yếu tố cốt lõi để trưởng thành trong học tập và nghiên cứu khoa học.

Chúng em xin được bày tỏ lòng biết ơn chân thành và sâu sắc nhất đến thầy Nguyễn Hữu Hiệp – người đã đồng hành, dẫn dắt và truyền cảm hứng cho chúng em trong suốt quá trình học tập. Không chỉ là người truyền đạt kiến thức, thầy còn là người gieo mầm tư duy logic, sự kỷ luật trong học thuật và niềm tin vào sức mạnh của tri thức. Chính sự tận tâm và tinh thần trách nhiệm trong từng buổi giảng của thầy đã tiếp thêm động lực để chúng em nghiêm túc hoàn thiện đề tài với tất cả sự nỗ lực và tôn trọng tri thức.

Chúng em hiểu rằng, kiến thức là hành trình không có điểm dừng. Dù đã dành nhiều tâm huyết cho đề tài, nhưng với giới hạn về thời gian và trải nghiệm, nhóm vẫn còn không ít thiếu sót. Rất mong thầy sẽ góp ý tận tình để chúng em có thể tiếp tục hoàn thiện hơn trên hành trình học thuật phía trước.

Một lần nữa, xin được gửi đến thầy lời cảm ơn sâu sắc cùng lời chúc sức khỏe, nhiệt huyết và thành công trên con đường trồng người cao quý.

Trân trọng

2 Tổng Quan Về Nén Dữ Liệu Bằng Phương Pháp SVD

2.1 Giới thiệu

Trong thời đại công nghệ số phát triển nhanh chóng, dữ liệu được tạo ra với tốc độ vượt bậc dưới nhiều dạng như hình ảnh, âm thanh, video, văn bản,... Tuy nhiên, không phải mọi dữ liệu đều mang giá trị như nhau - phần lớn có thể dư thừa hoặc lặp lại. Vì vậy, nén dữ liệu trở thành một giải pháp thiết yếu để tối ưu lưu trữ và truyền tải.

Trong đó, SVD (Singular Value Decomposition) là một phương pháp toán học mạnh mẽ, giúp phân tách dữ liệu thành các thành phần quan trọng nhất. Thay vì lưu toàn bộ dữ liệu gốc, ta chỉ giữ lại phần lõi - đủ để tái tạo lại một phiên bản nhẹ hơn nhưng vẫn đảm bảo nội dung chính. Khác với các phương pháp nén dựa trên mã hóa (như JPEG, MP3,...), SVD tiếp cận vấn đề theo hướng tuyến tính và cấu trúc ma trận, đặc biệt hiệu quả với ảnh, video và dữ liệu dạng bảng.

2.2 Lịch sử phát triển và Vai trò

2.2.1 Lịch sử hình thành và phát triển

Nguồn gốc của SVD có thể được truy về các công trình toán học cuối thế kỷ 19, khi nhà toán học người Anh *James Joseph Sylvester* và các đồng nghiệp bắt đầu nghiên cứu về phân rã ma trận. Tuy nhiên, phải đến năm 1936, hai nhà toán học *Carl Eckart* và *Gale Young* mới chính thức đưa ra lý thuyết đầy đủ về SVD và chứng minh rằng nó là công cụ tối ưu để xấp xỉ một ma trận bởi một ma trận có hạng thấp hơn.

Đến những năm 1965 - 1970, các nhà toán học như *Golub*, *Kahan*, và *Reinsch* phát triển những thuật toán tính toán SVD hiệu quả trên máy tính. Chính bước đột phá này đã biến SVD từ một công cụ lý thuyết trở thành một kỹ thuật ứng dụng được trong thực tiễn. Từ cuối thế kỷ 20 đến nay, SVD đã được ứng dụng rộng rãi trong:

- Xử lý tín hiệu và ảnh số,
- Phân tích dữ liệu lớn (Big Data),
- Trí tuệ nhân tạo và học máy,
- Phân tích ngôn ngữ tự nhiên,...

2.2.2 Vai trò trong thời đại dữ liệu số

Trong bối cảnh dữ liệu số ngày càng khổng lồ và đa dạng, SVD đóng vai trò then chốt trong việc:

- **Tối ưu hóa lưu trữ:** Giảm đáng kể dung lượng mà không làm mất đi giá trị thông tin.
- **Tăng hiệu quả truyền tải:** Dữ liệu nhẹ hơn giúp tiết kiệm băng thông và tăng tốc độ truyền.
- **Giảm chiều, khử nhiễu và làm mượt dữ liệu:** Góp phần làm đơn giản hóa mô hình học máy, tăng tốc độ xử lý, Giữ lại phần “cốt lõi” của dữ liệu, loại bỏ chi tiết nhiễu không cần thiết.

2.3 Phương pháp và ứng dụng nén dữ liệu sử dụng SVD

2.3.1 Phương pháp nén dữ liệu bằng SVD

Phân rã giá trị suy biến (SVD) là một phương pháp hiệu quả để nén dữ liệu thông qua việc phân tách ma trận dữ liệu thành các thành phần chính, từ đó loại bỏ thông tin dư thừa. Một số phương pháp

tiêu biểu bao gồm:

- **Nén ảnh số:** Áp dụng SVD lên ma trận độ sáng của ảnh và chỉ giữ lại các thành phần kỳ dị lớn nhất để tái tạo ảnh nén với chất lượng chấp nhận được.
- **Giảm chiều dữ liệu (Dimensionality Reduction):** Loại bỏ các chiều ít quan trọng trong dữ liệu nhằm giảm độ phức tạp khi xử lý, thường sử dụng trong học máy và trí tuệ nhân tạo.
- **Nén ma trận dữ liệu đa chiều:** Sử dụng trong các hệ thống xử lý dữ liệu lớn, chẳng hạn như ma trận tương tác người dùng – sản phẩm trong hệ thống gợi ý.
- **Kết hợp với kỹ thuật khác:** SVD có thể tích hợp với PCA, LSA, hoặc mạng nơ-ron sâu để tăng hiệu quả trích xuất đặc trưng và nén dữ liệu.

2.3.2 Ứng dụng thực tiễn

Các phương pháp nêu trên được ứng dụng rộng rãi trong nhiều lĩnh vực:

- **Xử lý ảnh và video:** Nén và phục hồi ảnh kỹ thuật số, ảnh y tế (X-quang, MRI), giảm nhiễu và giảm kích thước lưu trữ.
- **Phân tích văn bản và ngôn ngữ tự nhiên:** Sử dụng SVD kết hợp LSA để phân tích ngữ nghĩa, phân loại tài liệu, và tìm kiếm thông tin theo ngữ cảnh.
- **Học máy và trí tuệ nhân tạo:** Tiền xử lý dữ liệu giúp đơn giản hóa mô hình học, tăng tốc độ huấn luyện và giảm nguy cơ overfitting.
- **Viễn thông và cảm biến:** Truyền dữ liệu hiệu quả trong các mạng cảm biến với yêu cầu tiết kiệm năng lượng và độ trễ thấp.

3 Cơ Sở Lí Thuyết

3.1 Trị riêng và vector riêng

Cho một ma trận vuông $A \in \mathbb{R}^{n \times n}$, một số vô hướng λ và vector không rỗng $\mathbf{x} \in \mathbb{R}^n$ được gọi là **trị riêng (eigenvalue)** và **vector riêng (eigenvector)** tương ứng nếu thỏa mãn:

$$A\mathbf{x} = \lambda\mathbf{x}$$

Khi đó, λ là trị riêng của A , và \mathbf{x} là vector riêng tương ứng. Trị riêng biểu thị mức độ kéo giãn hoặc co lại theo hướng của vector riêng dưới tác động của phép biến đổi tuyến tính.

Một số tính chất:

- Nếu \mathbf{x} là vector riêng ứng với trị riêng λ thì mọi vector cùng phương với \mathbf{x} (tức $k\mathbf{x}$, với $k \neq 0$) cũng là vector riêng với cùng trị riêng.
- Mỗi ma trận vuông cấp n có thể có tối đa n trị riêng (tính cả bội đại số), có thể là số thực hoặc số phức.
- Nếu A là ma trận đối xứng thì tất cả các trị riêng đều là số thực, và có thể chọn các vector riêng tương ứng trực giao nhau.

- Nếu A là ma trận xác định dương, tất cả các trị riêng của nó đều là số thực dương; nếu A là nửa xác định dương, các trị riêng đều không âm.

3.2 Chéo hóa ma trận

Một ma trận vuông $A \in \mathbb{R}^{n \times n}$ được gọi là **chéo hóa được** nếu tồn tại ma trận khả nghịch P và ma trận đường chéo D sao cho:

$$A = PDP^{-1}$$

Trong đó:

- D là ma trận đường chéo chứa các trị riêng $\lambda_1, \lambda_2, \dots, \lambda_n$ của A trên đường chéo chính:

$$D = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}$$

- Ma trận $P = [\mathbf{p}_1 \ \mathbf{p}_2 \ \dots \ \mathbf{p}_n]$ là ma trận cột chứa các vector riêng tương ứng với các trị riêng λ_i .

Khi nhân hai vế với P , ta có:

$$AP = PD \Rightarrow A\mathbf{p}_i = \lambda_i \mathbf{p}_i$$

Điều kiện chéo hóa:

- Nếu ma trận A có n trị riêng phân biệt \Rightarrow chắc chắn chéo hóa được.
- Nếu không đủ vector riêng độc lập tuyến tính \Rightarrow không thể chéo hóa.
- Trong một số trường hợp đặc biệt, vẫn có thể đưa A về dạng chuẩn Jordan.

3.3 Hệ trực giao và trực chuẩn

Một hệ cơ sở $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_m \in \mathbb{R}^m$ được gọi là **trực giao** (orthogonal) nếu mỗi vector là khác 0 và tích của hai vector khác nhau bất kỳ bằng 0:

$$\mathbf{u}_i \neq 0, \quad \mathbf{u}_i^T \mathbf{u}_j = 0 \quad \forall 1 \leq i \neq j \leq m$$

Một hệ cơ sở $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_m \in \mathbb{R}^m$ được gọi là **trực chuẩn** (orthonormal) nếu nó là một hệ trực giao và độ dài Euclidean (chuẩn 2) của mỗi vector bằng 1:

$$\mathbf{u}_i^T \mathbf{u}_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Khi đó, tập hợp các vector trên tạo thành ma trận $U = [\mathbf{u}_1 \ \mathbf{u}_2 \ \dots \ \mathbf{u}_m]$, thỏa mãn:

$$U^T U = U U^T = I$$

Với I là ma trận đơn vị. Khi đó, U được gọi là **ma trận trực giao** (orthogonal matrix).

Tính chất của ma trận trực giao:

- $U^{-1} = U^T$: Nghịch đảo của ma trận trực giao chính là chuyển vị của nó.
- Nếu U là ma trận trực giao thì chuyển vị của nó U^T cũng là ma trận trực giao.
- $\det(U) = \pm 1$: Định thức của ma trận trực giao bằng 1 hoặc -1 .
- Giả sử $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$ và U là ma trận trực giao. Khi xoay:

$$(U\mathbf{x})^T(U\mathbf{y}) = \mathbf{x}^T U^T U \mathbf{y} = \mathbf{x}^T \mathbf{y}$$

do vậy phép xoay không làm thay đổi tích vô hướng giữa hai vector.

4 Phân Tích SVD (Singular Value Decomposition)

4.1 Singular Value Decomposition (SVD)

Phân tích giá trị đặc biệt (SVD) là một công cụ nền tảng trong đại số tuyến tính, cho phép phân rã bất kỳ ma trận thực $A \in \mathbb{R}^{m \times n}$ thành tích của ba ma trận có ý nghĩa hình học rõ ràng. Khác với phân tích trị riêng vốn chỉ áp dụng cho ma trận vuông, SVD có thể dùng cho mọi ma trận và được ứng dụng rộng rãi trong nén ảnh, khử nhiễu, phân tích dữ liệu, học máy và tối ưu hóa tuyến tính.

4.1.1 Biểu diễn tổng quát

Với một ma trận thực bất kỳ $A \in \mathbb{R}^{m \times n}$, luôn tồn tại ba ma trận:

- $U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times n}$: ma trận trực giao,
- $\Sigma \in \mathbb{R}^{m \times n}$: ma trận đường chéo (không nhất thiết vuông),

sao cho:

$$A_{m \times n} = U_{m \times m} \Sigma_{m \times n} (V_{n \times n})^T$$

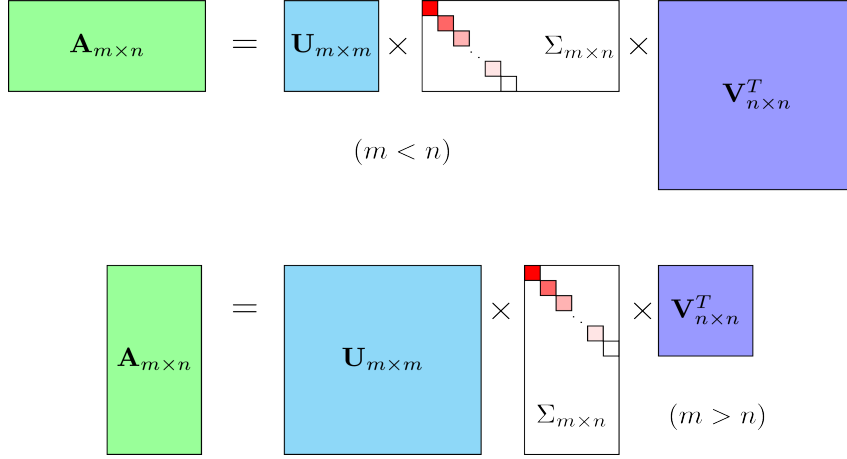
Trong đó:

- Các cột của U : *left-singular vectors* của A ,
- Các cột của V : *right-singular vectors* của A ,
- Ma trận Σ chứa các giá trị đặc biệt (singular values) $\sigma_1, \sigma_2, \dots, \sigma_r$ thỏa mãn:

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0, \quad \sigma_i = 0 \text{ với } i > r, \quad r = \text{rank}(A)$$

$$\Sigma = \begin{bmatrix} \sigma_1 & & & & \\ & \sigma_2 & & & \\ & & \ddots & & \\ & & & \sigma_r & \\ & & & & 0 \end{bmatrix}_{m \times n}$$

Các giá trị này nằm trên đường chéo chính của ma trận Σ , còn các phần tử ngoài đường chéo chính đều bằng 0, tạo thành một ma trận đường chéo ngay cả khi Σ không phải là ma trận vuông.



Hình 1: Minh họa phân rã SVD cho ma trận A trong hai trường hợp: $m < n$ (trên) và $m > n$ (dưới). Ma trận Σ là đường chéo, các giá trị giảm dần và không âm; màu đỏ đậm biểu thị giá trị lớn, ô trắng biểu thị giá trị bằng 0.

Lưu ý:

- Mặc dù ma trận Σ không nhất thiết là ma trận vuông, nhưng vẫn được xem là ma trận đường chéo với các phần tử khác 0 chỉ nằm trên đường chéo chính (vị trí có chỉ số hàng bằng chỉ số cột)
- Số lượng các phần tử khác 0 trong Σ chính là hạng của ma trận A , phản ánh số chiều độc lập tuyến tính trong ảnh của phép biến đổi tuyến tính tương ứng với A .

4.1.2 Mối liên hệ giữa SVD và phân tích trị riêng

Từ công thức phân rã SVD $A = U\Sigma V^T$, ta có thể suy ra một mối liên hệ quan trọng giữa SVD và phân tích trị riêng thông qua phép nhân AA^T . Cụ thể:

$$AA^T = U\Sigma V^T(U\Sigma V^T)^T = U\Sigma V^T V \Sigma^T U^T = U\Sigma \Sigma^T U^T$$

Ta sử dụng tính chất $V^T V = I$ do V là một ma trận trực giao. Như vậy, AA^T được phân tích dưới dạng $U\Sigma \Sigma^T U^T$, đây là dạng phân tích trị riêng (Eigen Decomposition) của ma trận AA^T .

Ma trận $\Sigma \Sigma^T$ là một ma trận đường chéo, với các phần tử trên đường chéo là $\sigma_1^2, \sigma_2^2, \dots$. Điều này cho thấy rằng các trị riêng của AA^T chính là bình phương các giá trị đặc biệt (singular values) của A , tức là:

$$\lambda_i = \sigma_i^2, \quad \text{với } i = 1, 2, \dots, r$$

Trong đó, $r = \text{rank}(A)$. Vì AA^T là một ma trận đối xứng và nửa xác định dương, nên các trị riêng của nó luôn là các số không âm. Do đó, các giá trị đặc biệt σ_i có thể được xác định bằng cách lấy căn bậc hai của các trị riêng tương ứng của AA^T .

Tương tự, bằng cách xét ma trận $A^T A$, ta cũng có:

$$A^T A = V \Sigma^T \Sigma V^T$$

Từ những phân tích trên, có thể thấy rằng phân tích giá trị đặc biệt không chỉ cung cấp một công cụ mạnh để biểu diễn và xử lý ma trận, mà còn làm sáng tỏ cấu trúc nội tại của ma trận thông qua liên hệ trực tiếp với các trị riêng của AA^T và $A^T A$.

Hệ quả:

- Các trị riêng của ma trận AA^T và $A^T A$ là các số không âm σ_i^2 .

- Các vector riêng của AA^T là các cột của U , được gọi là các *left-singular vectors* của A .
- Các vector riêng của $A^T A$ là các cột của V , được gọi là các *right-singular vectors* của A .
- Các giá trị σ_i là căn bậc hai của các trị riêng, được gọi là *giá trị đặc biệt* (singular values) của A .

4.1.3 Tính chất của phân tích SVD

- SVD luôn tồn tại với mọi ma trận thực A .
- Số lượng giá trị đặc biệt khác 0 chính là hạng của ma trận A .
- U và V là các ma trận trực giao: $U^T U = I_m$, $V^T V = I_n$.
- Phân rã SVD không duy nhất: có thể đổi dấu đồng thời các cặp vector tương ứng trong U và V .

4.2 Compact SVD (Rút Gọn SVD)

Trong phân rã SVD tiêu chuẩn, một ma trận bất kỳ $A \in \mathbb{R}^{m \times n}$ có thể được viết lại dưới dạng tổng các ma trận hạng 1 như sau:

$$A = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \cdots + \sigma_r \mathbf{u}_r \mathbf{v}_r^T$$

Trong đó:

- σ_i là các giá trị đặc biệt (singular values), sắp xếp giảm dần theo thứ tự $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$,
- $\mathbf{u}_i \in \mathbb{R}^m$ là các vector đơn vị trực giao (left singular vectors),
- $\mathbf{v}_i \in \mathbb{R}^n$ là các vector đơn vị trực giao (right singular vectors),
- $r = \text{rank}(A)$ là hạng của ma trận A .

Mỗi hạng tử $\sigma_i \mathbf{u}_i \mathbf{v}_i^T$ là một ma trận hạng 1, thể hiện một thành phần riêng biệt trong cấu trúc của A . Khi cộng tất cả các hạng tử này lại, ta thu được chính ma trận A . Nếu các giá trị σ_i giảm nhanh, ta có thể xấp xỉ A hiệu quả chỉ bằng một vài thành phần đầu tiên.

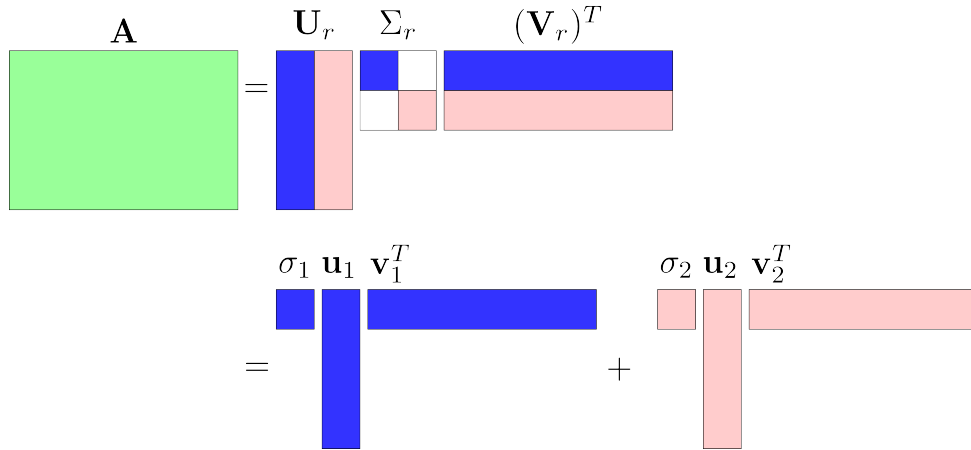
Tuy nhiên, trong nhiều trường hợp, việc lưu trữ đầy đủ cả ba ma trận $U \in \mathbb{R}^{m \times m}$, $\Sigma \in \mathbb{R}^{m \times n}$, và $V \in \mathbb{R}^{n \times n}$ là không cần thiết. Đặc biệt khi A có hạng thấp $r \ll \min(m, n)$, ta chỉ cần giữ lại r thành phần đầu tiên là đủ để khôi phục chính xác toàn bộ ma trận.

$$A = U_r \Sigma_r (V_r)^T$$

Trong đó:

- $U_r \in \mathbb{R}^{m \times r}$: chứa r vector đầu tiên (cột đầu tiên) của ma trận U ,
- $V_r \in \mathbb{R}^{n \times r}$: chứa r vector đầu tiên (cột đầu tiên) của ma trận V ,
- $\Sigma_r \in \mathbb{R}^{r \times r}$: là ma trận đường chéo chứa các giá trị đặc biệt $\sigma_1, \dots, \sigma_r$.

Compact SVD giữ nguyên tính chính xác của phân rã SVD ban đầu nhưng sử dụng ít bộ nhớ hơn và tăng tốc độ xử lý. Đây là một công cụ đặc biệt hữu ích trong các bài toán xử lý dữ liệu lớn, nén ảnh, và giảm chiều dữ liệu.



Hình 2: Biểu diễn SVD dạng thu gọn và biểu diễn ma trận dưới dạng tổng các ma trận có rank bằng 1.

4.3 Truncated SVD (Phép Xấp Xỉ SVD)

Trong phân rã SVD tiêu chuẩn của ma trận $A \in \mathbb{R}^{m \times n}$, các giá trị đặc biệt (singular values) trong ma trận Σ được sắp xếp giảm dần theo thứ tự:

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0 = \sigma_{r+1} = \dots = \sigma_{\min(m,n)}$$

Thông thường, chỉ một số ít giá trị σ_i đầu tiên là lớn đáng kể, trong khi phần còn lại nhỏ và gần bằng 0. Điều này cho phép ta xấp xỉ ma trận A bằng cách chỉ giữ lại $k < r$ thành phần đầu tiên — tức là chỉ giữ lại k giá trị đặc biệt lớn nhất cùng các vector tương ứng. Phép xấp xỉ này được gọi là **Truncated SVD** và được viết dưới dạng:

$$A \approx A_k = U_k \Sigma_k V_k^T = \sum_{i=1}^k \sigma_i \mathbf{u}_i \mathbf{v}_i^T = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \dots + \sigma_k \mathbf{u}_k \mathbf{v}_k^T$$

Trong đó:

- $U_k \in \mathbb{R}^{m \times k}$: chứa k vector riêng trái đầu tiên,
- $V_k \in \mathbb{R}^{n \times k}$: chứa k vector riêng phải đầu tiên,
- $\Sigma_k \in \mathbb{R}^{k \times k}$: chứa k giá trị đặc biệt đầu tiên.

Truncated SVD thường được sử dụng trong các ứng dụng như nén dữ liệu, giảm chiều dữ liệu, và loại bỏ nhiễu, vì nó giữ lại phần thông tin quan trọng nhất của ma trận mà không cần sử dụng toàn bộ SVD.

Định lý (Eckart–Young):

Truncated SVD bậc k là xấp xỉ tốt nhất của A theo chuẩn Frobenius. Sai số giữa A và A_k được cho bởi:

$$\|A - A_k\|_F^2 = \sum_{i=k+1}^r \sigma_i^2$$

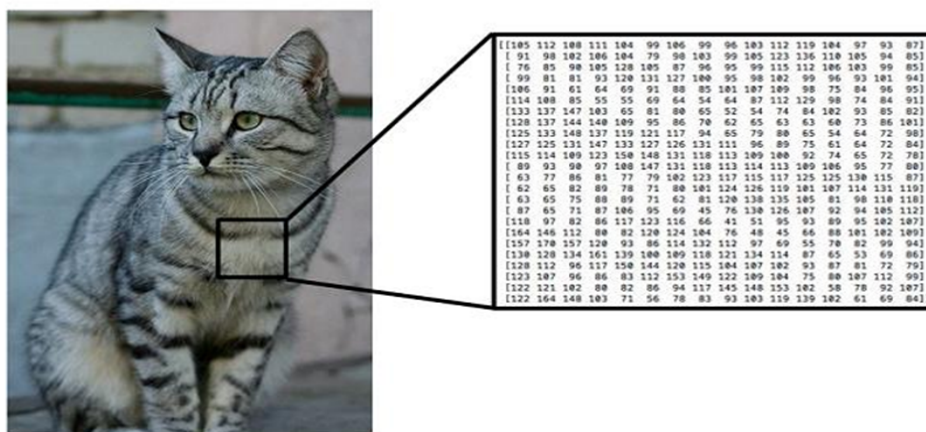
Điều này có nghĩa là tổng bình phương các giá trị đặc biệt bị bỏ qua chính là độ lệch giữa ma trận ban đầu và ma trận xấp xỉ. Nếu các σ_i sau k rất nhỏ, thì A_k sẽ gần đúng rất tốt với A .

5 Ứng Dụng SVD Để Nén Dữ Liệu

5.1 Phương thức lưu trữ dữ liệu hình ảnh trong máy tính

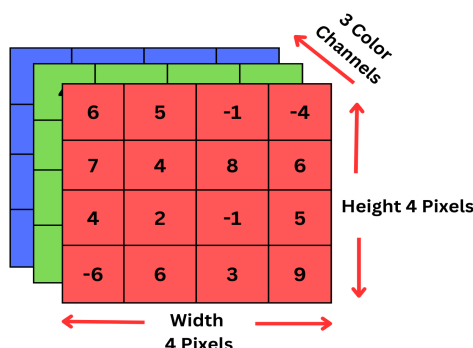
Trong hệ thống máy tính, mọi loại dữ liệu – từ văn bản, âm thanh cho đến hình ảnh – đều được số hóa và lưu trữ dưới dạng **mã nhị phân** gồm hai giá trị cơ bản là **0** và **1**. Đây là cơ sở hạ tầng cốt lõi cho phép máy tính tiếp nhận, xử lý và truyền tải thông tin.

Đối với hình ảnh kỹ thuật số, bản chất cấu trúc dữ liệu chính là một *tập hợp các điểm ảnh (pixel)*, những đơn vị cơ bản tạo nên toàn bộ khung hình. Mỗi pixel mang thông tin về **màu sắc** hoặc **mức độ sáng tối** và được mã hóa bằng một *giá trị số học*. Đối với hình ảnh màu, phổ biến nhất là hệ màu **RGB (Red – Green – Blue)**, trong đó mỗi kênh màu được biểu diễn bằng một số nguyên từ 0 đến 255. Tổ hợp ba giá trị này tạo nên màu sắc cụ thể tại mỗi pixel. Vì vậy, một hình ảnh màu có thể được biểu diễn bằng ba ma trận: ma trận kênh đỏ (R), xanh lục (G) và xanh lam (B). Với ảnh xám, chỉ một ma trận duy nhất biểu diễn mức xám (intensity) là đủ.



Hình 3: Minh họa ma trận điểm ảnh

Các ma trận này chính là biểu diễn số hóa của hình ảnh – tức là hình ảnh đã được ánh xạ về không gian toán học dưới dạng một tập hợp số liệu có cấu trúc. Từ đây, ta có thể lưu trữ chúng bằng các định dạng khác nhau: nhị phân (để tiết kiệm bộ nhớ), thập phân (dễ xử lý tính toán), hoặc mã hóa theo chuẩn hình ảnh như JPEG, PNG,...



Hình 4: Mô hình ba ma trận RGB

Chính cách biểu diễn hình ảnh dưới dạng ma trận đã tạo tiền đề cho việc ứng dụng các phương pháp xử lý số liệu tuyến tính trong xử lý ảnh. Một trong những kỹ thuật mạnh mẽ và phổ biến nhất là phân tích giá trị suy biến (Singular Value Decomposition – SVD). Phép phân tích SVD cho

phép *phân rã ma trận ảnh thành ba thành phần chính*, từ đó nắm bắt được những yếu tố quan trọng nhất của hình ảnh ban đầu. Việc này hỗ trợ nhiều mục tiêu như:

- **Nén ảnh:** Giảm kích thước lưu trữ nhưng vẫn duy trì chất lượng thị giác tốt.
- **Khử nhiễu:** Loại bỏ các tín hiệu không mong muốn mà vẫn giữ được chi tiết.
- **Tăng cường độ nét và độ tương phản:** Làm rõ những vùng mờ nhòe hoặc thiếu chi tiết.
- **Trích xuất đặc trưng ảnh:** Phục vụ cho các bài toán học máy, nhận diện khuôn mặt, thị giác máy tính, v.v.

Có thể nói, hình ảnh – dù là một thực thể mang tính trực quan – khi được mã hóa và phân tích toán học, đã trở thành một đối tượng định lượng có thể tính toán, tối ưu và khai thác một cách hiệu quả. Đó là nền tảng giúp xử lý ảnh không chỉ dừng lại ở mức hiển thị, mà còn tiến xa hơn trong các ứng dụng trí tuệ nhân tạo và khoa học dữ liệu hiện đại.

5.2 Xử lý dữ liệu hình ảnh bằng SVD

Trong lĩnh vực xử lý ảnh số, **phân tích giá trị suy biến** (Singular Value Decomposition – SVD) là một kỹ thuật nền tảng, được ứng dụng rộng rãi trong nén ảnh và giảm kích thước dữ liệu mà vẫn đảm bảo duy trì chất lượng thị giác. Với một ảnh xám được biểu diễn dưới dạng ma trận $A \in \mathbb{R}^{m \times n}$, ta có thể phân tích ma trận này thành tích của ba ma trận đặc biệt như sau:

$$A_{m \times n} = U_{m \times n} \Sigma_{m \times n} (V_{n \times n})^T$$

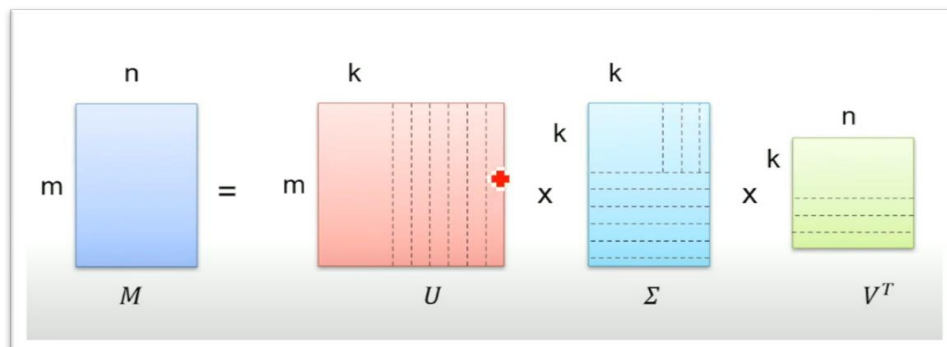
Thực tế cho thấy, hầu hết các ảnh số đều chứa một lượng lớn thông tin dư thừa. Điều này được phản ánh qua việc phần lớn các giá trị suy biến σ_i có độ lớn rất nhỏ hoặc gần bằng 0. Do đó, ta có thể xấp xỉ ma trận gốc bằng cách giữ lại chỉ k giá trị suy biến lớn nhất, với $k < r$. Khi đó, ảnh ban đầu được xấp xỉ bởi:

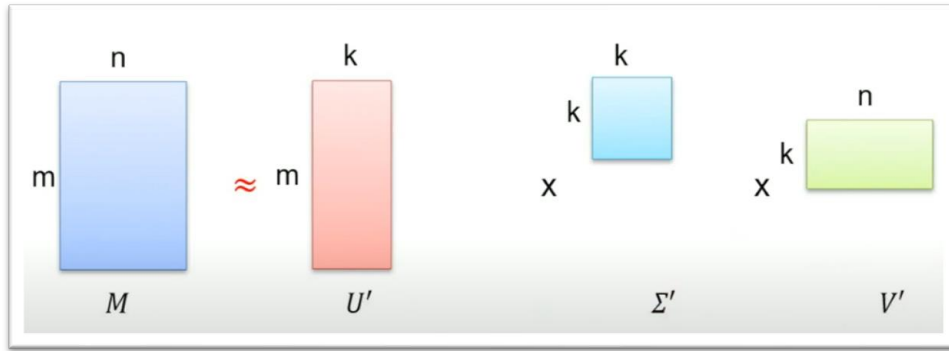
$$A \approx A_k = \sum_{i=1}^k \sigma_i u_i v_i^T$$

Hoặc viết lại dưới dạng ma trận:

$$A_k = U_k \Sigma_k V_k^T$$

Việc giữ lại một số lượng nhỏ giá trị suy biến giúp giảm thiểu đáng kể lượng dữ liệu cần lưu trữ, đồng thời vẫn giữ được phần lớn thông tin và cấu trúc ảnh gốc.





Hình 5: Minh họa quá trình loại bỏ các trị riêng trên đường chéo

Hiệu quả và ảnh hưởng đến chất lượng ảnh:

Phương pháp nén ảnh bằng SVD không chỉ mang lại hiệu quả lưu trữ vượt trội mà còn giúp điều chỉnh linh hoạt chất lượng ảnh tái tạo thông qua tham số k . Cụ thể:

- **Về mặt lưu trữ:** Thay vì cần lưu toàn bộ $m \times n$ phần tử của ảnh gốc, sau khi áp dụng SVD, ta chỉ cần lưu:

$$k(m + n + 1)$$

phần tử — bao gồm k cột đầu của U , k hàng đầu của V^T và k giá trị suy biến trong Σ . Điều này giúp giảm đáng kể kích thước dữ liệu, đặc biệt hiệu quả khi $k \ll \min(m, n)$.

- **Về chất lượng ảnh:** Lựa chọn k ảnh hưởng trực tiếp đến độ trung thực của ảnh tái tạo:
 - Nếu k quá nhỏ, chỉ giữ lại một phần rất ít thông tin quan trọng, ảnh sẽ bị mờ, mất chi tiết — đặc biệt là ở các vùng có kết cấu phức tạp.
 - Nếu k đủ lớn, ảnh tái tạo sẽ gần như không khác biệt với ảnh gốc về mặt thị giác.

Để cân bằng giữa hiệu quả nén và chất lượng, trong thực tiễn, người ta thường lựa chọn k sao cho tỷ lệ năng lượng giữ lại đạt từ 90% đến 95% so với tổng năng lượng ban đầu. Năng lượng ở đây được hiểu là tổng bình phương các giá trị suy biến, thể hiện mức độ đóng góp của từng thành phần trong việc khôi phục thông tin ảnh:

$$\frac{\sum_{i=1}^k \sigma_i^2}{\sum_{i=1}^r \sigma_i^2} \geq \text{Ngưỡng mong muốn} \quad (\text{ví dụ: } 0.90 \text{ hoặc } 0.95)$$

5.3 Tính toán dung lượng lưu trữ và tỷ lệ nén ảnh với Truncated SVD

Trong thực tế, sau khi áp dụng Truncated SVD để nén ảnh, thay vì lưu trữ toàn bộ ma trận ảnh gốc $A \in \mathbb{R}^{m \times n}$ với $m \times n$ phần tử, ta chỉ cần lưu ba ma trận đã rút gọn: $U_k \in \mathbb{R}^{m \times k}$ (gồm k cột đầu tiên của ma trận trực giao U), $\Sigma_k \in \mathbb{R}^{k \times k}$ (chứa k giá trị suy biến lớn nhất), và $V_k^T \in \mathbb{R}^{k \times n}$ (gồm k hàng đầu tiên của ma trận chuyển vị V^T).

Tổng số phần tử cần lưu là:

$$k(m + n + 1)$$

Trong đó, k phần tử cuối cùng tương ứng với k giá trị trên đường chéo chính của Σ_k .

- **Giả định định dạng lưu trữ:** Giả sử mỗi phần tử được lưu dưới dạng số thực 4 byte (tương đương kiểu float32), thì tổng số byte cần lưu trữ cho ảnh nén là:

$$4k(m + n + 1)(\text{byte})$$

- **So sánh với ảnh gốc:** Một ảnh gốc có kích thước $m \times n$ với mỗi điểm ảnh là một số nguyên 1 byte (tương đương `uint8`), tổng dung lượng ảnh gốc là:

$$m \times n \text{ (byte)}$$

- **Tỷ lệ nén:** Tỷ lệ nén r được định nghĩa là tỉ số giữa dung lượng ảnh nén và dung lượng ảnh gốc:

$$r = \frac{4k(m + n + 1)}{mn}$$

Từ công thức này, ta có thể biến đổi để tìm k tương ứng với một tỷ lệ nén mong muốn:

$$k = \frac{r \cdot mn}{4(m + n + 1)}$$

Ví dụ minh họa:

Xét một ảnh xám có kích thước 960×1440 ($m = 960$, $n = 1440$), chọn $k = 100$. Khi đó:

$$r = \frac{4 \cdot 100 \cdot (960 + 1440 + 1)}{960 \cdot 1440} = \frac{4 \cdot 100 \cdot 2401}{1382400} \approx 0.694$$

Nghĩa là ảnh nén chỉ chiếm khoảng 69.4% dung lượng so với ảnh gốc — tức tiết kiệm được 30.6% bộ nhớ, trong khi vẫn giữ lại hầu hết thông tin thị giác quan trọng. Đây là minh chứng rõ ràng cho hiệu quả của Truncated SVD trong việc giảm dung lượng lưu trữ mà không làm giảm đáng kể chất lượng ảnh.

6 Code Python

6.1 Giới thiệu Python

Python là một ngôn ngữ lập trình thông dịch bậc cao, được phát triển bởi Guido van Rossum và ra mắt lần đầu vào năm 1991. Với cú pháp đơn giản, dễ đọc, Python nhanh chóng trở thành một trong những ngôn ngữ phổ biến nhất trên thế giới, đặc biệt trong các lĩnh vực như trí tuệ nhân tạo, khoa học dữ liệu, thị giác máy tính, phát triển web và tự động hóa.

Python là một ngôn ngữ đa năng, hỗ trợ nhiều mô hình lập trình như hướng đối tượng, hàm và thủ tục. Triết lý thiết kế của Python nhấn mạnh vào tính rõ ràng và dễ bảo trì, giúp người học dễ tiếp cận và triển khai ứng dụng thực tế.

Một số đặc điểm nổi bật của Python gồm:

- Cú pháp đơn giản, gần với ngôn ngữ tự nhiên.
- Hệ sinh thái thư viện phong phú như `numpy`, `opencv`, `matplotlib`, `pandas`, `scikit-learn`.
- Tính linh hoạt cao, chạy được trên nhiều hệ điều hành.

6.2 Các hàm thường dùng trong Python

Trong Python, có rất nhiều hàm và thư viện được sử dụng phổ biến để xử lý dữ liệu, tính toán, vẽ biểu đồ và xử lý ảnh. Dưới đây là một số nhóm hàm và thư viện thường dùng, được trình bày theo bảng kèm ví dụ minh họa.

Hàm	Công dụng	Ví dụ
print()	In ra màn hình	print("Hello")
len()	Trả về độ dài đối tượng	len("abc") → 3
int(), float(), str()	Ép kiểu dữ liệu cơ bản	int("5") → 5
type()	Kiểm tra kiểu dữ liệu	type(3.14) → float
range()	Tạo dãy số	list(range(5)) → [0,1,2,3,4]

Bảng 2: Một số hàm cơ bản trong Python

Hàm	Công dụng	Ví dụ
np.array()	Tạo mảng NumPy	a = np.array([1,2,3])
np.zeros()	Tạo ma trận 0	np.zeros((2,2))
np.dot()	Nhân ma trận	np.dot(A, B)
np.shape	Trả về kích thước ma trận	A.shape → (2, 2)
np.linalg.inv()	Ma trận nghịch đảo	np.linalg.inv(A)
np.copy()	Sao chép mảng gốc sang mảng mới	B = np.copy(A)
np.outer()	Tích ngoài hai vector	np.outer(a, b)

Bảng 3: Một số hàm trong thư viện NumPy

Hàm	Công dụng	Ví dụ
cv2.imread()	Đọc ảnh từ file	img = cv2.imread("a.jpg")
cv2.imshow()	Hiển thị ảnh	cv2.imshow("Ảnh", img)
cv2.cvtColor()	Chuyển đổi màu sắc	cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2.imwrite()	Lưu ảnh	cv2.imwrite("b.jpg", img)

Bảng 4: Một số hàm trong thư viện OpenCV

Hàm	Công dụng	Ví dụ
plt.plot()	Vẽ biểu đồ đường	plt.plot([1,2,3], [4,5,6])
plt.imshow()	Hiển thị ảnh	plt.imshow(img)
plt.title()	Đặt tiêu đề hình	plt.title("Tiêu đề")
plt.show()	Hiển thị đồ thị	plt.show()
plt.semilogy()	Vẽ đồ thị log trực y (thường cho giá trị SVD)	plt.semilogy(sigma)

Bảng 5: Một số hàm trong thư viện Matplotlib

6.3 Code python

```

1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 # Đọc ảnh và chuyển sang ảnh xám
6 I = cv2.imread('Man.jpg', cv2.IMREAD_GRAYSCALE)

```

```
7 [m, n] = I.shape
8
9 # Hien thi anh goc
10 plt.figure(1)
11 plt.imshow(I, cmap='gray')
12 plt.title('Anh goc')
13 plt.axis('off')
14
15 # Chuyen anh sang kieu float (gia tri thuc tu 0 den 1)
16 I = I.astype(np.float64) / 255.0
17
18 # Phan ra SVD
19 U, S, Vt = np.linalg.svd(I, full_matrices=False)
20 sigma = S.copy() # cac gia tri suy bien
21
22 # Tinh anh tai to hop voi thanh phan dau tien
23 I1 = sigma[0] * np.outer(U[:, 0], Vt[0, :])
24
25 # Nhap k tu nguoi dung
26 k = int(input("Nhap vao k (so luong thanh phan chinh): "))
27
28 # Tai to hop anh voi k thanh phan dau tien
29 for i in range(1, k):
30     I1 += sigma[i] * np.outer(U[:, i], Vt[i, :])
31
32 # Hien thi anh tai to hop
33 plt.figure(2)
34 plt.imshow(I1, cmap='gray')
35 plt.title(f'Anh tai to hop voi k = {k}')
36 plt.axis('off')
37
38 # Ve do thi cac gia tri singular
39 plt.figure(3)
40 plt.semilogy(sigma, 'r.-')
41 plt.ylabel('Singular values')
42 plt.title('Bieu do gia tri thanh phan chinh')
43 plt.grid(True)
44
45 plt.show()
```

6.4 Kết quả chạy code

Đoạn mã Python ở phần trước áp dụng kỹ thuật phân rã giá trị suy biến (SVD – Singular Value Decomposition) để giảm nhiễu ảnh xám. Cụ thể, ảnh Man.jpg được đọc và chuyển sang ảnh xám, sau đó phân tích thành các thành phần chính. Quá trình tái tổ hợp ảnh được thực hiện bằng cách giữ lại k thành phần chính đầu tiên, nơi k do người dùng nhập vào.

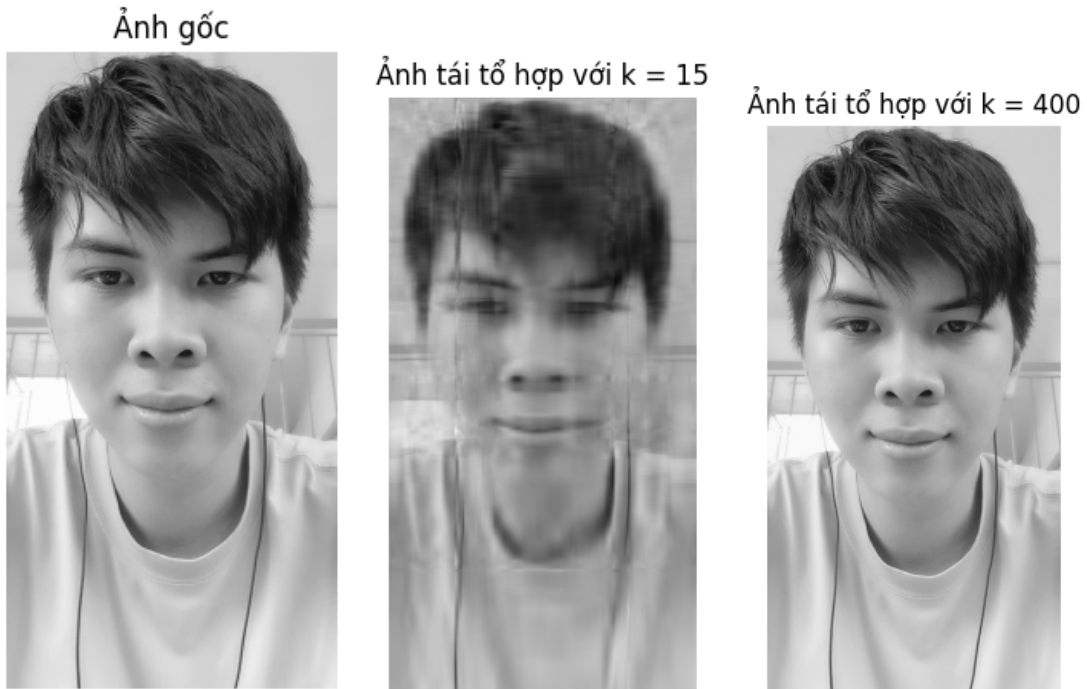
Đây là kết quả thị giác khi lựa chọn các giá trị k khác nhau:

Trường hợp $k = 15$:

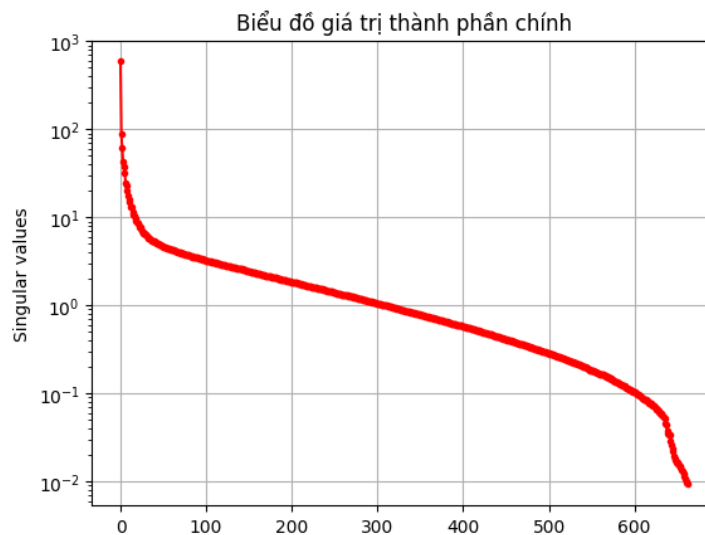
- Ảnh được tái tổ hợp từ 15 thành phần chính đầu tiên.
- Cho ra mức độ nén rất cao, dung lượng ảnh giảm mạnh.
- Tuy nhiên, ảnh bị mất chi tiết nghiêm trọng, xuất hiện mờ, nhòe, mất đường nét rõ ràng.
- Chỉ phù hợp cho mục đích nén mạnh, không dùng được nếu yêu cầu chất lượng cao.

Trường hợp $k = 400$:

- Ảnh tái tổ hợp giữ lại 400 thành phần, gần tương đương ảnh gốc.
- Chất lượng ảnh gần như nguyên vẹn, chi tiết rõ ràng, không còn nhòe.
- Dung lượng ảnh gần bằng ảnh ban đầu, hiệu quả nén thấp.
- Phù hợp khi yêu cầu giữ chất lượng cao, không quan trọng dung lượng.



Hình 6: So sánh ảnh gốc và ảnh tái tổ hợp theo k



Hình 7: Giá trị của các singular values của ma trận ảnh theo k

Kết luận

Giá trị k – số lượng thành phần chính được giữ lại – đóng vai trò then chốt trong việc cân bằng giữa chất lượng ảnh và mức độ nén:

- Khi k quá nhỏ \Rightarrow ảnh bị suy giảm nghiêm trọng về mặt thị giác: mất chi tiết, mờ nhòe, không sử dụng được cho các mục đích yêu cầu chất lượng cao.
- Khi k quá lớn \Rightarrow ảnh gần như không nén được bao nhiêu, làm mất đi ưu điểm của phương pháp nén bằng SVD.

Vì vậy, cần lựa chọn giá trị k tối ưu, sao cho vừa đảm bảo chất lượng ảnh tái tạo đủ tốt, vừa giúp giảm dung lượng hiệu quả, tùy vào yêu cầu cụ thể của từng ứng dụng.

Tài liệu

- [1] Đặng Văn Vinh. (20/7/2025). *Đại số tuyến tính*. Trường Đại học Bách khoa, Đại học Quốc gia TP. Hồ Chí Minh.
- [2] Stanford University. (20/7/2025). *Singular value decomposition*. Retrieved from <https://www.youtube.com/watch?v=P5mlg91as1c>
- [3] Princeton University. (20/7/2025). *Singular value decomposition*. Retrieved from <https://www.cs.princeton.edu/courses/archive/spring12/cos598C/svdchapter.pdf>
- [4] Stanford University. (20/7/2025). *CS168: The modern algorithmic toolbox, lecture 9: The singular value decomposition (SVD) and low-rank matrix approximations*. Retrieved from <https://theory.stanford.edu/~tim/s15/l/19.pdf>