

ĐẠI HỌC QUỐC GIA
THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN



MTH00057 - Toán ứng dụng và thống kê cho Công
nghệ thông tin

BÁO CÁO ĐỒ ÁN 1

Color Compression

Họ tên MSSV
Bùi Minh Duy 23127040

Giảng viên hướng dẫn
Nguyễn Văn Quang Huy
Trần Hà Sơn
Nguyễn Đình Thúc
Nguyễn Ngọc Toàn

Ngày 23 tháng 6 năm 2025

Mục lục

1	Ý tưởng thực hiện	2
1.1	Tổng quan	2
1.2	Input, Output	2
1.3	Ý tưởng giải quyết	2
2	Chi tiết thực hiện	3
2.1	Cấu trúc chương trình	3
2.2	Chi tiết thuật toán K-Means	3
2.3	Tối ưu hiệu năng	3
3	Kết quả và kết luận	4
3.1	Chất lượng hình ảnh nén theo số màu	4
3.2	Thời gian thực thi	7
3.3	Số vòng lặp hội tụ	8
3.4	So sánh hai phương pháp khởi tạo centroid	9
3.5	Giới hạn và vấn đề gặp phải	11
3.6	Kết luận	11
4	Tài liệu tham khảo	12
5	Acknowledgement	12

1 Ý tưởng thực hiện

1.1 Tổng quan

Trong bài toán này, mục tiêu là giảm số lượng màu trong ảnh gốc nhưng vẫn giữ lại được càng nhiều chi tiết và chất lượng càng tốt. Việc giảm số lượng màu giúp tiết kiệm dung lượng lưu trữ và đơn giản hóa quá trình xử lý ảnh.

Phương pháp được sử dụng là thuật toán K-Means Clustering, một phương pháp phân nhóm (clustering) phổ biến trong lĩnh vực học máy không giám sát. Thuật toán sẽ gom nhóm các pixel màu tương đồng lại thành k cụm (clusters), và đại diện cho mỗi cụm là một màu centroid duy nhất.

1.2 Input, Output

- **Input:** Một file ảnh màu (dạng PNG, JPG, hoặc BMP)
- **Output:** Ảnh mới đã được giảm số màu xuống k màu tùy chọn, lưu dưới dạng PNG và PDF.

1.3 Ý tưởng giải quyết

Ảnh màu là một ma trận 3 chiều kích thước ($height, width, 3$), mỗi pixel có giá trị màu RGB. Ý tưởng của thuật toán:

1. Chuyển ảnh thành dạng ma trận 1D gồm N điểm dữ liệu, mỗi điểm có 3 thuộc tính (R, G, B)
2. Khởi tạo k centroid ngẫu nhiên (hoặc lấy trực tiếp từ pixel của ảnh)
3. Tính khoảng cách từ từng pixel đến tất cả các centroid
4. Gán mỗi pixel vào cụm có centroid gần nhất
5. Tính lại các centroid mới là trung bình các pixel thuộc mỗi cụm
6. Lặp lại các bước trên cho đến khi các centroid hội tụ (không thay đổi đáng kể) hoặc đạt số lần lặp tối đa
7. Dùng các centroid thu được để gán lại màu cho từng pixel, tái tạo ảnh mới với k màu

Thuật toán này đơn giản, dễ cài đặt, và đặc biệt phù hợp với bài toán nén số lượng màu trong ảnh thực tế.

2 Chi tiết thực hiện

2.1 Cấu trúc chương trình

Chương trình gồm các hàm chính sau:

- **read_img**: Đọc ảnh từ đường dẫn cho trước và trả về ma trận ảnh 2D
- **show_img**: Hiển thị ảnh sử dụng thư viện matplotlib
- **save_img**: Lưu ảnh với định dạng PNG hoặc PDF
- **convert_img_to_1d**: Chuyển ma trận ảnh 2D về dạng ma trận 1D ($N, 3$)
- **kmeans**: Thực hiện thuật toán K-Means Clustering
- **generate_2d_img**: Tái tạo ảnh 2D từ centroids và labels sau khi phân cụm

2.2 Chi tiết thuật toán K-Means

Bước 1: Khởi tạo centroid

- Nếu chọn ‘random’: khởi tạo ngẫu nhiên giá trị RGB từ 0–255
- Nếu chọn ‘in_pixels’: chọn ngẫu nhiên k pixel từ ảnh làm centroid

Bước 2: Gán pixel vào cụm gần nhất

Sử dụng numpy broadcasting để tính khoảng cách Euclidean (bình phương) giữa tất cả các pixel và centroid. Sau đó, gán mỗi pixel vào cụm có centroid gần nhất.

Bước 3: Cập nhật centroid

Tính lại centroid mới là trung bình cộng các pixel thuộc cùng một cụm.

Bước 4: Kiểm tra hội tụ

Lặp lại bước 2 và 3 cho đến khi centroid mới gần bằng centroid cũ với sai số tuyệt đối nhỏ hơn $1e - 3$ (0.001) hoặc đã đạt số vòng lặp tối đa.

2.3 Tối ưu hiệu năng

Để tăng tốc thuật toán:

- Sử dụng numpy broadcasting thay cho vòng lặp for
- Tính khoảng cách bình phương để tránh căn bậc hai
- Kiểm tra hội tụ sớm bằng `np.allclose`

3 Kết quả và kết luận

Tiến hành thực nghiệm thuật toán K-Means với các giá trị k khác nhau để giảm số màu của ảnh, đồng thời phân tích chất lượng kết quả nén dựa trên các yếu tố như chất lượng ảnh trực quan, thời gian thực thi và số vòng lặp hội tụ.

3.1 Chất lượng hình ảnh nén theo số màu

Kết quả ảnh nén với số lượng màu $k = 3, 5, 7, 9$ được thể hiện dưới đây, đồng thời so sánh trực quan giữa hai phương pháp khởi tạo centroid: *random* và *in_pixels*.



Hình 1: Ảnh gốc dùng để so sánh nén màu.



(a) random



(b) in_pixels

Hình 2: Ảnh nén với $k = 3$



(a) random



(b) in_pixels

Hình 3: Ảnh nén với $k = 5$



(a) random



(b) in_pixels

Hình 4: Ảnh nén với $k = 7$



(a) random



(b) in_pixels

Hình 5: Ảnh nén với $k = 9$

Nhận xét chất lượng ảnh nén theo số màu

- **Với $k = 3$:** Ảnh chỉ giữ lại ba màu chủ đạo nên nhiều chi tiết nhỏ bị mất, đặc biệt ở các vùng chuyển màu. Tuy nhiên, các mảng màu lớn vẫn được phân biệt rõ và bối cảnh tổng thể của ảnh vẫn giữ nguyên.
- **Với $k = 5$:** Hình ảnh trở nên rõ ràng hơn, các gam màu phụ bắt đầu xuất hiện, giúp tái hiện các vùng chuyển tiếp màu và chi tiết nhỏ tốt hơn so với mức $k = 3$.
- **Với $k = 7$:** Ảnh gần như giữ lại đầy đủ các thông tin thị giác quan trọng. Các vùng chuyển tiếp màu mượt mà, giảm thiểu hiện tượng phân mảng màu, chất lượng hình ảnh tương đương với ảnh gốc về khả năng nhận diện.
- **Với $k = 9$:** Chất lượng tiếp tục được cải thiện nhẹ nhưng không quá khác biệt so với $k = 7$. Trong khi đó, chi phí tính toán tiếp tục tăng, cho thấy $k = 7$ đã là mức tối ưu cho bài toán này.

Ngoài ra, so sánh giữa hai phương pháp khởi tạo centroid:

- *in_pixels* có xu hướng tái hiện màu trung thực hơn ở các mức k thấp do chọn trực tiếp các giá trị từ pixel thực tế.
- *random* đôi khi gây sai lệch nhẹ về màu sắc ở mức k thấp, nhưng khi k tăng, sự khác biệt giữa hai phương pháp này dần thu hẹp và kết quả gần tương đồng.

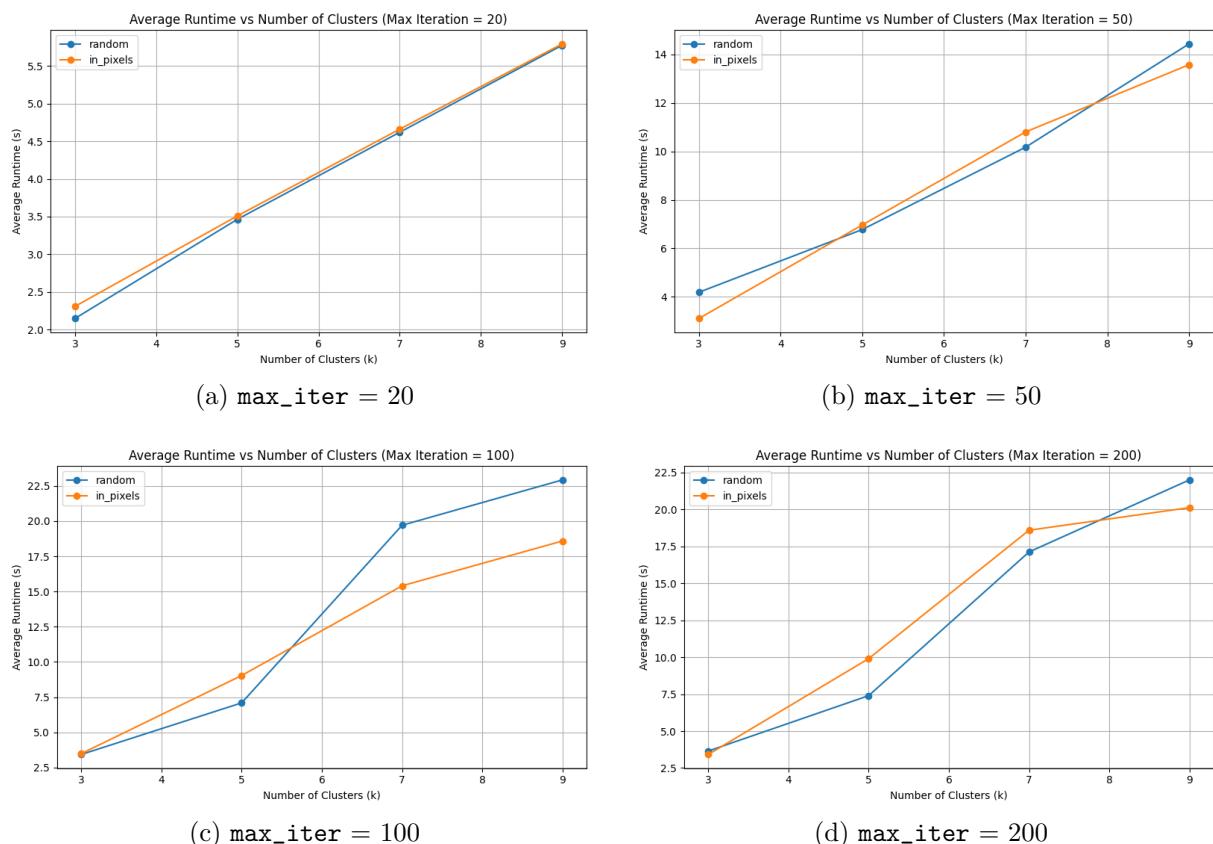
Kết luận: Chất lượng ảnh nén tăng rõ rệt theo số cụm màu k , nhưng từ $k = 7$ trở đi, sự cải thiện thị giác không còn nhiều trong khi thời gian tính toán tăng đáng kể. Do đó, giá trị $k = 7$ được xem là hợp lý nhất cho bài toán nén màu ảnh RGB kích thước trung bình.

3.2 Thời gian thực thi

Kết quả benchmark cho thấy thời gian thực thi của thuật toán k -means chịu ảnh hưởng trực tiếp bởi hai tham số chính: số cụm k và số vòng lặp tối đa `max_iter`.

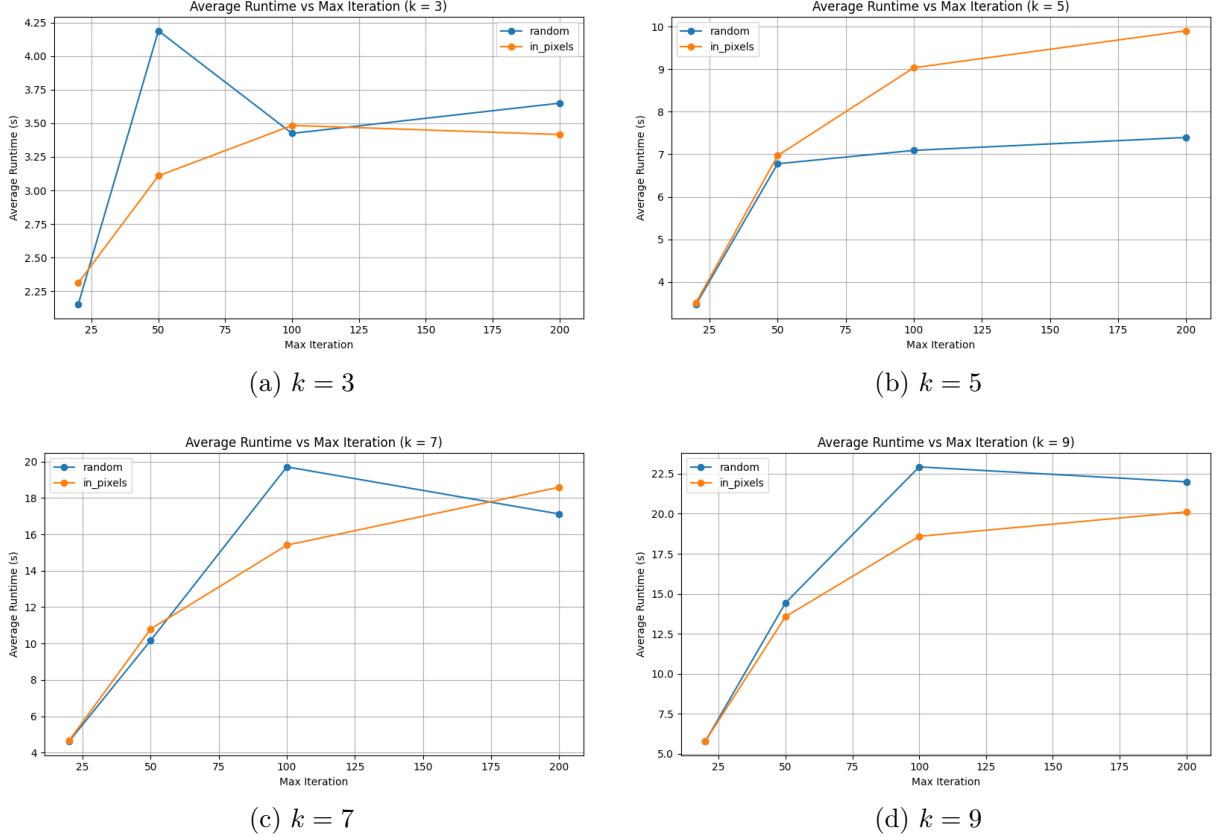
- Khi tăng số cụm k , số lượng phép tính khoảng cách và cập nhật centroid ở mỗi vòng lặp tăng lên, dẫn đến thời gian thực thi dài hơn.
- Khi tăng `max_iter`, nếu thuật toán chưa hội tụ sớm, tổng số vòng lặp thực hiện sẽ nhiều hơn, làm tăng thời gian xử lý.

Ảnh hưởng của `max_iter` được thể hiện trong Hình 6. Để thấy rằng thời gian thực thi tăng dần khi `max_iter` tăng từ 20 lên 200, đặc biệt rõ rệt khi số cụm k lớn.



Hình 6: Thời gian thực thi trung bình với các giá trị `max_iter` khác nhau

Ảnh hưởng của số cụm k được thể hiện rõ qua Hình 7. Khi tăng từ 3 lên 9 cụm, thời gian xử lý tăng gần tuyến tính, do mỗi vòng lặp cần tính nhiều khoảng cách hơn và quá trình cập nhật centroid cũng phức tạp hơn.



Hình 7: Thời gian thực thi trung bình với các giá trị số cụm k khác nhau

3.3 Số vòng lặp hội tụ

Qua quá trình benchmark với các giá trị `max_iter` khác nhau (20, 50, 100, 200) và số cụm k đa dạng, có thể rút ra các nhận xét sau về số vòng lặp hội tụ của thuật toán k-means:

- Với `max_iter` thấp (20): hầu hết trường hợp không thể hội tụ trước khi đạt giới hạn. Điều này cho thấy 20 vòng thường quá ít để thuật toán ổn định với dữ liệu ảnh và các cấu hình k thông thường.
- Khi tăng `max_iter` lên khoảng 50: tỉ lệ hội tụ trước ngưỡng tăng lên so với 20, nhưng vẫn còn nhiều trường hợp đặc biệt với k lớn hoặc khởi tạo không thuận lợi không kịp hội tụ. Các lần hội tụ (nếu có) thường ở khoảng 20–50 vòng.
- Với `max_iter` ở mức 100: đa số trường hợp hội tụ trước ngưỡng, nhưng số vòng thực tế dao động rộng (có thể từ 20 đến gần 100), phụ thuộc vào k và cách khởi tạo centroid. Điều này cho thấy 100 vòng là mức hợp lý để hầu hết hội tụ, nhưng vẫn cần quan sát độ dao động của số vòng thực tế.
- Khi `max_iter` đạt 200: phần lớn các trường hợp hội tụ, kể cả với k lớn, nhưng đôi khi số vòng cần thiết rất cao (gần 200). Do đó, mặc dù tăng ngưỡng giúp nâng khả năng hội tụ, chi phí tính toán cũng tăng đáng kể khi thuật toán cần nhiều vòng.
- Xu hướng chung: số vòng lặp hội tụ tăng theo k . Với k nhỏ, thường hội tụ nhanh

hơn; khi k lớn, cần nhiều vòng để centroid ổn định. Phương pháp khởi tạo cũng ảnh hưởng: khởi tạo từ pixel thực (`in_pixels`) có thể giúp hội tụ nhanh hơn ở một số cấu hình, nhưng không hoàn toàn ổn định cho mọi k .

- Khuyến nghị: để đảm bảo hội tụ, cần chọn `max_iter` đủ cao (ít nhất 100) với ảnh trung bình; đồng thời nên cân nhắc khởi tạo thông minh (ví dụ k-means++) để giảm số vòng lặp thực tế.

3.4 So sánh hai phương pháp khởi tạo centroid

Dưới đây là phân tích dựa trên kết quả benchmark (thời gian trung bình) cho các kết hợp `max_iter` = {20, 50, 100, 200} và số cụm k = {3, 5, 7, 9}. Bảng dữ liệu trung bình về thời gian thực thi (giây) như sau:

Bảng 1: Thời gian thực thi trung bình (`avg_time`) theo phương pháp khởi tạo centroid, cho từng `max_iter` và k .

<code>max_iter</code>	k	<code>random</code>	<code>in_pixels</code>	Phương pháp nhanh hơn	Ghi chú
20	3	2.149	2.309	random	random nhanh hơn rõ
	5	3.465	3.509		chênh lệch nhỏ
	7	4.620	4.661		chênh lệch rất nhỏ
	9	5.773	5.792		chênh lệch rất nhỏ
50	3	4.189	3.109	<code>in_pixels</code>	<code>in_pixels</code> nhanh hơn đáng kể
	5	6.776	6.967		random nhanh hơn nhẹ
	7	10.172	10.801		random nhanh hơn
	9	14.431	13.572		<code>in_pixels</code> nhanh hơn
100	3	3.425	3.485	random	chênh lệch rất nhỏ
	5	7.090	9.032		random nhanh hơn rõ
	7	19.706	15.407		<code>in_pixels</code> nhanh hơn đáng kể
	9	22.930	18.590		<code>in_pixels</code> nhanh hơn rõ
200	3	3.650	3.416	<code>in_pixels</code>	<code>in_pixels</code> nhanh hơn nhẹ
	5	7.395	9.902		random nhanh hơn đáng kể
	7	17.120	18.589		random nhanh hơn nhẹ
	9	21.995	20.118		<code>in_pixels</code> nhanh hơn

Từ bảng trên, rút ra một số nhận xét chính:

- Với `max_iter` = 20 (giới hạn thấp):
 - Ở mọi $k \in \{3, 5, 7, 9\}$, `random` đều nhanh hơn `in_pixels`, dù chênh lệch có khi rất nhỏ.
- Với `max_iter` = 50 (vừa phải):
 - Ở $k = 3$ và $k = 9$, `in_pixels` nhanh hơn do khởi tạo ban đầu gần giá trị pixel thực, giúp hội tụ sớm.

- Ở $k = 5$ và $k = 7$, `random` vẫn nhỉnh hơn một chút, cho thấy khi k trung bình, khởi tạo ngẫu nhiên đôi khi phân bố centroid hợp lý hơn.

- **Với `max_iter = 100` (cao hơn):**

- Ở $k = 3$, hai phương pháp gần tương đương.
- Ở $k = 5$, `random` nhanh hơn rõ.
- Ở $k = 7$ và $k = 9$, `in_pixels` nhanh hơn đáng kể, do khả năng khởi tạo gần vùng màu thực giúp giảm số vòng lặp khi k lớn.

- **Với `max_iter = 200` (rất cao):**

- Ở $k = 3$, `in_pixels` nhanh hơn nhẹ.
- Ở $k = 5$ và $k = 7$, `random` nhanh hơn, cho thấy khởi tạo ngẫu nhiên giúp thuật toán tránh bị kẹt khi có nhiều màu trung bình/phức tạp.
- Ở $k = 9$, `in_pixels` nhanh hơn, do khởi tạo từ dữ liệu thực giúp hội tụ nhanh trong không gian màu rất đa dạng.

Kết luận tổng quát

- Không có phương pháp khởi tạo nào *luôn* vượt trội trong mọi cấu hình.
- `in_pixels` có xu hướng *ổn định và nhanh hơn* khi k rất nhỏ (3) hoặc rất lớn (9) ở một số trường hợp (50, 100, 200).
- `random` thường chiếm ưu thế ở k trung bình (5, 7) trong nhiều trường hợp, nhất là khi `max_iter` không quá thấp, nhờ khả năng phân bố centroid ban đầu rộng hơn, tránh khởi tạo trùng vùng màu nhiều.
- Khi `max_iter` quá thấp (20), `random` thường nhanh hơn vì thuật toán có ít vòng để điều chỉnh, và khởi tạo ngẫu nhiên đôi khi may mắn phân bố tốt hơn.
- Tính ổn định: `in_pixels` ít dao động hơn với các k rất nhỏ hoặc rất lớn, nhưng với k trung bình có thể bị khởi tạo trùng vùng màu, làm chậm. `random` dao động mạnh hơn nhưng không dễ bị kẹt lâu ở k trung bình.

Gợi ý thực tiễn

- Với ảnh có phân bố màu *đơn giản*, *ít màu* hoặc khi cần số cụm rất nhỏ ($k = 3$), ưu tiên `in_pixels` để hội tụ nhanh.
- Với ảnh có *màu trung bình/phức tạp* (số lượng màu và sắc độ tương đối đa dạng) và k trung bình (5–7), `random` thường hiệu quả hơn.
- Với k rất lớn (≥ 9) và giới hạn vòng lặp đủ cao, `in_pixels` có thể cho thời gian thực thi tốt hơn, nhưng cần lưu ý kiểm tra tính ổn định (nên chạy nhiều lần với seed khác nhau).

- Luôn cân nhắc sử dụng khởi tạo k-means++ trong thực tế để giảm rủi ro khởi tạo kém và cải thiện hội tụ chung cho mọi k .

3.5 Giới hạn và vấn đề gặp phải

Trong quá trình thực nghiệm và benchmark, một số giới hạn và vấn đề sau được ghi nhận:

- **Quy mô và đa dạng của tập ảnh:** Bộ benchmark sử dụng số lượng hạn chế (khoảng 10 ảnh) với độ phân giải đã resize về mức cố định. Chưa đánh giá được hiệu năng với tập ảnh lớn, độ phân giải rất cao hoặc đa dạng hoàn cảnh nhiều, ánh sáng, làm giảm tính khái quát của kết quả.
- **Khả năng hội tụ không đảm bảo ở ngưỡng thấp:** Với `max_iter` thấp (ví dụ 20), đa phần trường hợp không hội tụ kịp, cho thấy ngưỡng vòng lặp cần đủ cao ($\geq 50\text{--}100$) để đảm bảo thuật toán ổn định với nhiều cấu hình k .
- **Độ nhạy với khởi tạo centroid:** Cả hai phương pháp (`random`, `in_pixels`) đôi khi gặp trường hợp khởi tạo không thuận lợi, dẫn tới số vòng lặp lớn hoặc không hội tụ trong giới hạn. Điều này cho thấy cần khởi tạo thông minh hơn (ví dụ k-means++)
- **Chi phí tính toán khi k và `max_iter` cao:** Khi số cụm k tăng (≥ 7) và `max_iter` lên đến 100–200, thời gian thực thi tăng rất nhanh, đôi khi tới vài chục giây cho mỗi ảnh. Điều này đặt áp lực về tài nguyên và thời gian, nhất là khi xử lý nhiều ảnh hoặc độ phân giải cao.

3.6 Kết luận

Qua quá trình thực nghiệm và phân tích kết quả, có thể rút ra một số kết luận chính về thuật toán K-Means trong nén màu ảnh:

- **Ảnh hưởng của tham số k và `max_iter`:** Thời gian thực thi và số vòng lặp hội tụ tăng rõ khi số cụm k lớn (≥ 7) và khi ngưỡng `max_iter` đủ cao (≥ 100). Ngưỡng quá thấp (20) thường không đảm bảo hội tụ, ngưỡng ở mức vừa phải (50) cải thiện nhưng chưa chắc đã hội tụ với mọi cấu hình, và mức $\geq 100\text{--}200$ là cần thiết để đa số trường hợp hội tụ.
- **So sánh khởi tạo centroid:**
 - Phương pháp `in_pixels` tỏ ra hội tụ nhanh hơn và ổn định hơn trong nhiều trường hợp với k rất nhỏ (3) hoặc rất lớn (9) khi ngưỡng vòng lặp đủ cao, nhờ khởi tạo gần giá trị pixel thực.
 - Phương pháp `random` linh hoạt hơn với k trung bình (5–7) và có thể tránh việc chọn centroid ban đầu trùng vùng màu nhiều, nhưng đôi khi cần nhiều vòng lặp hơn hoặc không hội tụ kịp ở ngưỡng thấp.
 - Cả hai cách đều có nhược điểm khi khởi tạo kém, do đó không có phương pháp nào luôn tối ưu tuyệt đối cho mọi loại ảnh.
- **Ưu tiên khởi tạo thông minh:** Kết quả nhấn mạnh tầm quan trọng của phương

pháp khởi tạo như *k-means++*, giúp phân bố centroid ban đầu hiệu quả hơn, giảm số vòng lặp và tăng xác suất hội tụ.

- **Cân bằng hiệu năng và chất lượng:** Trong thực tế, cần cân nhắc chọn k hợp lý (thường 5–9 cho nhiều bài toán nén màu) và đặt `max_iter` đủ cao (≥ 100) để đảm bảo hội tụ, đồng thời kiểm soát chi phí tính toán (có thể giảm độ phân giải hoặc trích mẫu điểm).
- **Tối ưu triển khai:** Để áp dụng trên tập lớn hoặc ảnh độ phân giải cao, nên kết hợp song song hóa, vectorization.

4 Tài liệu tham khảo

1. Wikipedia, [k-means clustering](#), truy cập ngày 15/06/2025.
2. GeeksforGeeks, [K means Clustering - Introduction](#), truy cập ngày 15/06/2025.
3. Machine Learning cơ bản, [Bài 4: K-means Clustering](#), truy cập ngày 20/06/2025.

5 Acknowledgement

Trong quá trình tối ưu thuật toán và hoàn thiện báo cáo, đồ án có tham khảo một số ý tưởng, đoạn mã mẫu và hướng dẫn từ ChatGPT, cụ thể trong phần tối ưu thuật toán K-Means bằng NumPy broadcasting, kiểm tra hội tụ sớm bằng `np.allclose`