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

BÁO CÁO

ĐỒ ÁN 01: COLOR COMPRESSION

Môn học: Toán ứng dụng và thống kê cho Công nghệ thông tin - 22CLC10

Giáo viên hướng dẫn:

Thầy Vũ Quốc Hoàng

Thầy Nguyễn Văn Quang Huy

Thầy Nguyễn Ngọc Toàn

Cô Phan Thị Phương Uyên

Sinh viên thực hiện:

22127440 Phan Võ Minh Tuệ

Thành phố Hồ Chí Minh - 2024

MỤC LỤC

LỜI CẢM ƠN

Trước hết, tôi xin bày tỏ lòng biết ơn sâu sắc đến quý thầy cô giáo hướng dẫn, đặc biệt là cô Phan Thị Phương Uyên, người đã trao cho tôi cơ hội thực hiện đồ án này. Đây quả thực là một chủ đề nghiên cứu hết sức thú vị nhưng cũng mang nhiều ý nghĩa quan trọng đối với tôi trên con đường chinh phục bộ môn Khoa học máy tính sau này.

Tôi cũng muốn gửi lời cảm ơn chân thành đến các bạn Huỳnh Gia Khánh, Trương Hoài Nam, Lâm Chí Tài và Nguyễn Quốc Tín đã cung cấp thông tin, chia sẻ kinh nghiệm và tiếp thêm động lực cho tôi trong suốt thời gian thực hiện đồ án lần này.

Cuối cùng, tôi không thể không nhắc đến gia đình yêu quý của mình, những người luôn ở bên cạnh, hỗ trợ và tạo điều kiện tốt nhất để tôi có thể tập trung vào việc học tập và nghiên cứu.

Lòng biết ơn của tôi cũng dành cho tất cả những ai đã giúp đỡ, trực tiếp hay gián tiếp, trong quá trình hoàn thành tiểu luận này.

Phan Võ Minh Tuệ

TỔNG QUAN

Đồ án 01 - Color Compression này nhằm mục đích giảm số lượng màu trong ảnh bằng cách sử dụng phương pháp gom nhóm màu với thuật toán K-Means. Mã nguồn được viết bằng ngôn ngữ lập trình Python, kết hợp với các thư viện NumPy, PIL và matplotlib để thực hiện các chức năng như đọc và hiển thị ảnh, xử lý dữ liệu, áp dụng thuật toán K-Means, và tái tạo ảnh mới từ các màu trung tâm đã được gom nhóm.

Nội dung chi tiết

Báo cáo sẽ trình bày chi tiết về cách thức thực hiện từng chức năng, bao gồm:

- Đọc và hiển thị ảnh: Sử dụng PIL để đọc và matplotlib để hiển thị ảnh.
- Chuyển đổi ảnh: Sử dụng NumPy để chuyển đổi ảnh thành dạng mảng 1D phù hợp cho K-Means.
- Gom nhóm màu: Áp dụng thuật toán K-Means để gom nhóm màu, khởi tạo và cập nhật các centroid.
- Tạo và lưu ảnh mới: Tạo ảnh mới từ các màu trung tâm và lưu ảnh đã nén vào tệp.
- Đánh giá kết quả: So sánh kết quả với các số lượng màu khác nhau để đánh giá hiệu suất và chất lượng của ảnh đã nén.

Kết quả

Các kết quả thử nghiệm sẽ được minh họa bằng hình ảnh và biểu đồ, cùng với nhận xét về hiệu suất của thuật toán K-Means trong việc giảm số lượng màu của ảnh. Điều này cung cấp cái nhìn tổng quan và sâu sắc về việc sử dụng thuật toán K-Means trong xử lý ảnh.

I. Ý TƯỞNG THỰC HIỆN VÀ MÔ TẢ CÁC HÀM

1. Ý tưởng thực hiện

- Sử dụng thuật toán **Mini Batch K-Means** (một phiên bản cải tiến của thuật toán K-Means truyền thống) để giảm số lượng màu trong ảnh nhằm giảm dung lượng cần thiết để lưu trữ ảnh. Ý tưởng chính của thuật toán là các pixel của ảnh sẽ được chia thành **k** nhóm dựa trên màu sắc. Mỗi nhóm được đại diện bởi một **centroid** (trung tâm) và màu sắc của mỗi pixel được thay thế bằng màu sắc của centroid của nhóm mà nó thuộc về.

- Các bước thực hiện:

1. Đọc ảnh từ đường dẫn.
2. Chuyển đổi ảnh từ kích thước 2D (cao, rộng, số chiều) sang 1D (cao * rộng, số chiều).
3. Khởi tạo các centroid.
4. Sử dụng Mini Batch K-Means để gán nhãn và cập nhật centroid cho đến khi hội tụ hoặc đạt số lần lặp tối đa.
5. Gán nhãn cuối cùng cho tất cả các pixel dựa trên các centroid.
6. Tạo ảnh mới từ các centroid và nhãn.
7. Lưu ảnh mới với màu sắc đã giảm.

- Lí do chọn thuật toán Mini Batch K-Means:

- Hiệu quả tính toán: Mini Batch K-Means giảm chi phí tính toán bằng cách sử dụng một tập con ngẫu nhiên của dữ liệu trong mỗi lần cập nhật centroid, giúp tăng tốc độ và hiệu quả, đặc biệt với dữ liệu lớn.
- Khả năng mở rộng: So với K-Means truyền thống, Mini Batch K-Means xử lý tốt hơn với các tập dữ liệu lớn vì không cần lặp lại trên toàn bộ dữ liệu, chỉ cần một lô ngẫu nhiên trong mỗi lần lặp.
- Độ chính xác: Dù có thể không chính xác bằng K-Means truyền thống, Mini Batch K-Means vẫn cung cấp kết quả tốt và đủ cho nhiều ứng dụng thực tế, bao gồm giảm số lượng màu trong ảnh.
- Tiết kiệm bộ nhớ: Bằng cách sử dụng các lô dữ liệu ngẫu nhiên, Mini Batch K-Means giảm yêu cầu về bộ nhớ, hữu ích khi làm việc với dữ liệu lớn mà không thể lưu trữ toàn bộ trong bộ nhớ.

2. Các thư viện được sử dụng

❖ **NumPy** là một thư viện mạnh mẽ để xử lý các mảng và các phép toán số học. Trong đoạn mã này, NumPy được sử dụng để:

- Chuyển đổi và thao tác trên các mảng hình ảnh.

- Tính toán khoảng cách Euclidean giữa các điểm ảnh và các centroid.

- Khởi tạo và cập nhật các centroid trong thuật toán K-means.

- Tạo các batch nhỏ để tăng tốc độ tính toán K-means.

❖ **PIL (Pillow)** là một thư viện để mở, xử lý và lưu trữ các định dạng hình ảnh khác nhau. Trong đoạn mã này, Pillow được sử dụng để:

- Đọc hình ảnh từ tệp vào mảng NumPy.

- Chuyển đổi mảng NumPy trở lại thành hình ảnh.

- Lưu trữ hình ảnh đã nén vào các tệp đầu ra.

❖ **Matplotlib** là một thư viện để tạo các biểu đồ và hình ảnh trực quan. Trong đoạn mã này, Matplotlib được sử dụng để:

- Hiển thị hình ảnh gốc và các hình ảnh đã nén trên các subplots.

- Đặt tiêu đề cho các hình ảnh để dễ dàng so sánh và đánh giá.

❖ **Time** là một thư viện để cung cấp các hàm liên quan đến thời gian. Trong đoạn mã này, time được sử dụng để đo thời gian thực hiện của thuật toán K-Means cho từng giá trị của k_clusters.

❖ **OS** là một thư viện để cung cấp một cách thuận tiện để làm việc với hệ thống tệp và thư mục. Trong đoạn mã này, os được sử dụng để:

- Kiểm tra sự tồn tại của tệp hình ảnh đầu vào.

- Tạo các thư mục để lưu trữ các hình ảnh đã nén.

3. Mô tả các hàm

❖ **read_img(img_path: str) → np.ndarray**

- Chức năng: Đọc hình ảnh từ đường dẫn và chuyển đổi nó thành mảng NumPy hai chiều để biểu diễn.

- Giải thích: Sử dụng thư viện PIL để mở và đọc hình ảnh, sau đó chuyển đổi nó thành mảng NumPy để dễ dàng xử lý bằng các thuật toán số.

❖ **show_img(ax: matplotlib.axes.Axes, img_2d: np.ndarray, title: str) → None**

- Chức năng: Hiển thị hình ảnh trên một subplot của matplotlib.

- Giải thích: Sử dụng matplotlib để hiển thị hình ảnh với tiêu đề, và ẩn các trục tọa độ để tập trung vào hình ảnh.

❖ **save_img(img_2d: np.ndarray, img_path: str) → None**

- Chức năng: Lưu hình ảnh dưới dạng tệp.

- Giải thích: Chuyển đổi mảng NumPy thành đối tượng hình ảnh và lưu nó vào đường dẫn chỉ định.

❖ **convert_img_to_1d(img_2d: np.ndarray) → np.ndarray**

- Chức năng: Chuyển đổi hình ảnh 2D thành không gian 1D.

- Giải thích: Thay đổi hình dạng của mảng NumPy từ (height, width, channels) thành (height * width, channels) hay còn gọi là làm phẳng ảnh để dễ dàng xử lý bằng thuật toán K-Means.

❖ **initialize_centroids(img_1d: np.ndarray, k_clusters: int, init_centroids: str) → np.ndarray**

- Chức năng: Khởi tạo các centroid ban đầu cho thuật toán K-Means.

- Giải thích: Có hai phương pháp khởi tạo centroid:

+ Chọn ngẫu nhiên từ không gian màu RGB (random), mỗi giá trị kênh nằm trong khoảng từ 0 đến 255.

+ Chọn ngẫu nhiên các điểm ảnh từ hình ảnh gốc (in_pixels).

- **assign_clusters(img_1d: np.ndarray, centroids: np.ndarray) → np.ndarray**

- Chức năng: Gán mỗi điểm ảnh vào cụm gần nhất dựa trên khoảng cách Euclidean.

- Giải thích: Tính khoảng cách từ mỗi điểm ảnh đến tất cả các centroid, sau đó gán nó vào cụm có khoảng cách nhỏ nhất.

- **update_centroids(img_1d: np.ndarray, labels: np.ndarray, k_clusters: int) → np.ndarray**

- Chức năng: Cập nhật vị trí các centroid dựa trên trung bình của các điểm ảnh trong cụm.

- Giải thích: Tính trung bình của tất cả các điểm ảnh trong mỗi cụm để cập nhật vị trí centroid. Nếu một cụm không có điểm ảnh nào, khởi tạo lại centroid ngẫu nhiên.

- **kmeans(img_1d: np.ndarray, k_clusters: int, max_iter: int, init_centroids: str = 'random') → Tuple[np.ndarray, np.ndarray]**

- Chức năng: Thực hiện thuật toán K-means để phân cụm các điểm ảnh.

- Giải thích: Khởi tạo các centroid, sau đó lặp lại quá trình gán cụm và cập nhật centroid cho đến khi hội tụ hoặc đạt số lần lặp tối đa. Sử dụng batch nhỏ để tăng tốc độ tính toán.

- **generate_2d_img(img_2d_shape: Tuple[int, int, int], centroids: np.ndarray, labels: np.ndarray) → np.ndarray**

- Chức năng: Tạo lại hình ảnh 2D từ các centroid và nhãn cụm.

- Giải thích: Sử dụng centroid của từng cụm để tạo lại hình ảnh từ không gian 1D về không gian 2D.

- **main() → None**

- Chức năng: Hàm chính thực hiện quy trình nén ảnh bằng cách sử dụng thuật toán K-Means, từ việc đọc ảnh, chuyển đổi ảnh, khởi tạo centroid, gán nhãn, cập nhật centroid, tạo và lưu ảnh mới.

II. HÌNH ẢNH KẾT QUẢ

- **Môi trường thực thi:** Google Colab

- **Giá trị biến mặc định:**

    + Số lần lặp tối đa: 100

    + Batch size: 1000

- **Hướng dẫn thực thi mã nguồn:**

1. Truy cập Google Colab

2. Tải file mã nguồn lên *[file đính kèm]*.

3. Tải hình ảnh mẫu thử lên.

4. Chạy mã nguồn. (Thay đổi loại *runtime*. Nếu không, mặc định sẽ là chạy bằng CPU)

5. Chạy hàm **main** và nhập vào các thông tin cần thiết để chạy như: đường dẫn đến ảnh cần thực thi, chọn số lượng màu sau khi nén màu, chọn định dạng được lưu (PNG, JPG hoặc PDF).

6. Xem và lưu kết quả.

1. Mẫu thử 1

Ảnh gốc

- Kích thước: 728 x 725
- Dung lượng: 239 KB
- Số lượng màu: 244397

Ảnh tạo bởi Phương pháp khởi tạo centroid 'Random'

- Dung lượng: 108 KB
- Số lượng màu: 3
- Thời gian: 0.1499s

- Dung lượng: 142 KB
- Số lượng màu: 5
- Thời gian: 0.1846s

- Dung lượng: 146 KB
- Số lượng màu: 7
- Thời gian: 0.2352s

Ảnh tạo bởi Phương pháp khởi tạo centroid 'In Pixels'

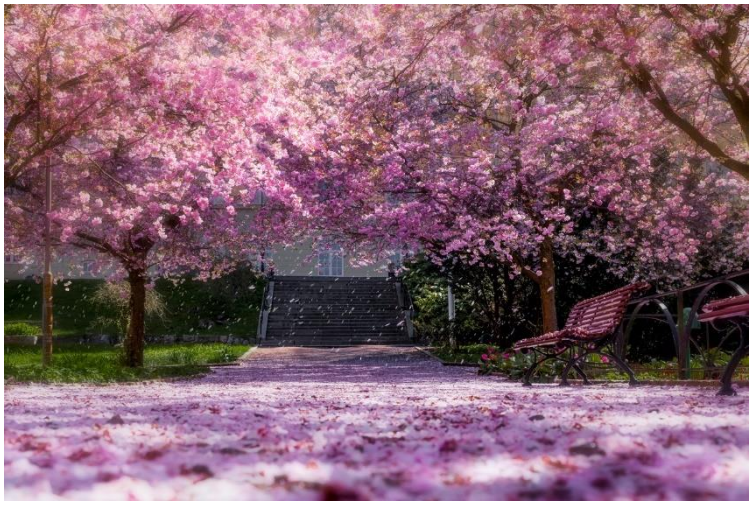
- Dung lượng: 110 KB
- Số lượng màu: 3
- Thời gian: 0.1619s

- Dung lượng: 145 KB
- Số lượng màu: 5
- Thời gian: 0.2075s

- Dung lượng: 138 KB
- Số lượng màu: 7
- Thời gian: 0.2768s

2. Mẫu thử 2

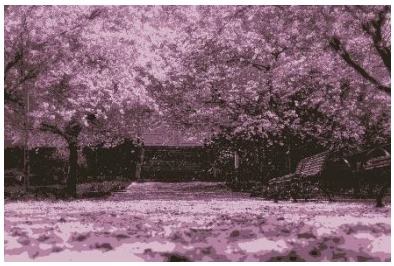
Ảnh gốc



- Kích thước: 1254 x 836
- Dung lượng: 1.16 MB
- Số lượng màu: 209002

Ảnh tạo bởi Phương pháp khởi tạo centroid 'Random'

- Dung lượng: 290 KB
- Số lượng màu: 3
- Thời gian: 0.2732s



- Dung lượng: 294 KB
- Số lượng màu: 5
- Thời gian: 0.3827s

- Dung lượng: 286 KB
- Số lượng màu: 7
- Thời gian: 0.5810s

Ảnh tạo bởi Phương pháp khởi tạo centroid 'In Pixels'

- Dung lượng: 296 KB
- Số lượng màu: 3
- Thời gian: 0.2805s

- Dung lượng: 294 KB
- Số lượng màu: 5
- Thời gian: 0.4036s

- Dung lượng: 298 KB
- Số lượng màu: 7
- Thời gian: 0.5419s

3. Mẫu thử 3

Ảnh gốc

- Kích thước: 3840 x 2160
- Dung lượng: 5.98 MB
- Số lượng màu: 754660

Ảnh tạo bởi Phương pháp khởi tạo centroid 'Random'

- Dung lượng: 1.15 MB
- Số lượng màu: 3
- Thời gian: 3.1097s

- Dung lượng: 1.30 MB
- Số lượng màu: 5
- Thời gian: 4.6603s

- Dung lượng: 1.38 MB
- Số lượng màu: 7
- Thời gian: 6.0230s

Ảnh tạo bởi Phương pháp khởi tạo centroid 'In Pixels'

- Dung lượng: 1.15 MB
- Số lượng màu: 3
- Thời gian: 2.2322s

- Dung lượng: 1.29 MB
- Số lượng màu: 5
- Thời gian: 3.5303s

- Dung lượng: 1.40 MB
- Số lượng màu: 7
- Thời gian: 6.3503s

III. NHẬN XÉT KẾT QUẢ

Dựa trên kết quả thu được từ phần trên, chúng ta có thể rút ra các nhận xét sau về chất lượng màu sắc, dung lượng giảm, và thời gian xử lý của chúng như sau:

1. **Mẫu thử 1**

❖ **Phương pháp khởi tạo centroid 'Random':**

K = 3	K = 5	K = 7
Dung lượng giảm còn 108 KB (giảm 2.2 lần)	Dung lượng giảm còn 142 KB (giảm 1.7 lần)	Dung lượng giảm còn 146 KB (giảm 1.6 lần)
Thời gian xử lý: 0.1499s	Thời gian xử lý: 0.1846s	Thời gian xử lý: 0.2352s

❖ **Phương pháp khởi tạo centroid 'In Pixels':**

K = 3	K = 5	K = 7
Dung lượng giảm còn 110 KB (giảm 2.2 lần)	Dung lượng giảm còn 145 KB (giảm 1.6 lần)	Dung lượng giảm còn 138 KB (giảm 1.7 lần)
Thời gian xử lý: 0.1619s	Thời gian xử lý: 0.2075s	Thời gian xử lý: 0.2768s

Nhận xét: Mẫu thử 1 có kích thước và dung lượng ban đầu nhỏ, nên sự khác biệt về thời gian xử lý giữa các phương pháp không quá lớn. Phương pháp 'Random' và 'In Pixels' đều cho thấy giảm dung lượng khá tốt nhưng chất lượng màu sắc giảm mạnh khi số lượng màu thấp (3 màu). Với 5 và 7 màu, ảnh vẫn giữ được độ phân giải tốt, mặc dù không hoàn toàn như ảnh gốc nhưng vẫn giữ được khá tốt nội dung tấm ảnh.

2. **Mẫu thử 2**

❖ **Phương pháp khởi tạo centroid 'Random':**

K = 3	K = 5	K = 7
Dung lượng giảm còn 290 KB (giảm 4 lần)	Dung lượng giảm còn 294 KB (giảm 3.9 lần)	Dung lượng giảm còn 286 KB (giảm 4.1 lần)

Thời gian xử lý: 0.2732s	Thời gian xử lý: 0.3827s	Thời gian xử lý: 0.5810s

❖ **Phương pháp khởi tạo centroid 'In Pixels':**

K = 3	K = 5	K = 7
Dung lượng giảm còn 296 KB (giảm 3.9 lần)	Dung lượng giảm còn 294 KB (giảm 3.9 lần)	Dung lượng giảm còn 298 KB (giảm 3.9 lần)
Thời gian xử lý: 0.2805s	Thời gian xử lý: 0.4036s	Thời gian xử lý: 0.5419s

Nhận xét: Mẫu thử 2 có dung lượng và kích thước lớn hơn, do đó sự khác biệt về thời gian xử lý giữa hai phương pháp cũng lớn hơn. Phương pháp 'In Pixels' thường cho thời gian xử lý nhanh hơn, đặc biệt khi số lượng màu tăng lên. Dung lượng giảm đáng kể với cả hai phương pháp, nhưng chất lượng màu sắc bị ảnh hưởng nhiều nhất khi chỉ còn 3 màu. Với 5 và 7 màu, ảnh vẫn giữ được độ phân giải tốt, nhưng không hoàn toàn như ảnh gốc.

3. **Mẫu thử 3**

❖ **Phương pháp khởi tạo centroid 'Random':**

K = 3	K = 5	K = 7
Dung lượng giảm còn 1.15 MB (giảm 5.2 lần)	Dung lượng giảm còn 142 KB (giảm 1.7 lần)	Dung lượng giảm còn 146 KB (giảm 1.6 lần)
Thời gian xử lý: 3.1097s	Thời gian xử lý: 4.6603s	Thời gian xử lý: 6.0230s

❖ **Phương pháp khởi tạo centroid 'In Pixels':**

K = 3	K = 5	K = 7
Dung lượng giảm còn 110 KB (giảm 2.2 lần)	Dung lượng giảm còn 145 KB (giảm 1.6 lần)	Dung lượng giảm còn 138 KB (giảm 1.7 lần)
Thời gian xử lý: 2.2322s	Thời gian xử lý: 3.5303s	Thời gian xử lý: 6.3503s

Nhận xét: Mẫu thử 3 có dung lượng và kích thước lớn nhất, do đó sự khác biệt về thời gian xử lý giữa hai phương pháp cũng rõ rệt hơn. Phương pháp 'In Pixels' thường cho thời gian xử lý nhanh hơn, đặc biệt khi số lượng màu cao. Dung lượng giảm đáng kể với cả hai phương pháp, nhưng chất lượng màu sắc bị ảnh hưởng nhiều nhất khi chỉ còn 3 màu. Với 5 và 7 màu, ảnh vẫn giữ được độ phân giải tốt, nhưng không hoàn toàn như ảnh gốc.

4. **Tổng kết chi tiết**

- **Chất lượng màu sắc:** Khi số lượng màu giảm xuống còn 3, chất lượng màu sắc bị giảm rõ rệt, nhưng khi sử dụng 5 và 7 màu, chi tiết màu sắc vẫn được giữ khá tốt.

- **Dung lượng giảm:** Cả hai phương pháp đều cho kết quả giảm dung lượng đáng kể, từ 1.6 đến 5.2 lần so với ảnh gốc, tùy thuộc vào số lượng màu được giữ lại.

- **Thời gian xử lý:** Phương pháp 'In Pixels' thường cho thời gian xử lý nhanh hơn so với 'Random', đặc biệt là khi số lượng màu cao. Tuy nhiên, cả hai phương pháp đều tăng thời gian xử lý khi số lượng màu tăng lên. Mini Batch K-Means đặc biệt hiệu quả trong việc xử lý các ảnh lớn hơn, giảm thời gian xử lý một cách đáng kể so với K-Means truyền thống.

- **Tính hiệu quả của Mini Batch K-Means:** Thuật toán Mini Batch K-Means cho thấy hiệu quả vượt trội trong việc xử lý các tập dữ liệu lớn nhờ giảm thời gian xử lý mà vẫn giữ được chất lượng ảnh tốt. Đây là một giải pháp tốt khi cần cân bằng giữa chất lượng màu sắc và hiệu quả xử lý.

TỔNG KẾT

Đồ án Color Compression đã thành công trong việc áp dụng thuật toán K-Means để giảm số lượng màu của các ảnh đầu vào. Qua quá trình thực hiện, tôi đã đạt được các mục tiêu đề ra như sau:

1. Đọc, hiển thị và lưu ảnh: Chương trình có khả năng xử lý các tệp ảnh đầu vào, hiển thị ảnh gốc và lưu các ảnh đã được nén lại sau khi áp dụng thuật toán K-Means.
2. Chuyển đổi ảnh và gom nhóm màu: Sử dụng thư viện PIL và NumPy để chuyển đổi ảnh thành định dạng phù hợp cho việc áp dụng K-Means. Sau đó, thuật toán K-Means được thực hiện để gom nhóm các màu tương tự trong ảnh.
3. Tạo ảnh mới và đánh giá kết quả: Bằng cách tái tạo lại ảnh từ các màu trung tâm đã gom nhóm, tôi đã giảm số lượng màu so với ảnh gốc. Quá trình thử nghiệm với các giá trị khác nhau của số lượng màu cho phép tôi đánh giá và so sánh chất lượng của các ảnh nén.
4. Nhận xét về hiệu suất: Kết quả đồ án đã minh họa rõ ràng sự hiệu quả của thuật toán K-Means trong việc giảm số lượng màu cũng như dung lượng của bức ảnh mà vẫn giữ được phần lớn nội dung của hình ảnh.

Trong tương lai, để cải thiện đồ án này, tôi có thể cân nhắc mở rộng phạm vi áp dụng thuật toán cho các loại hình ảnh và thêm các phương pháp phân tích kết quả nén để đánh giá độ chính xác và hiệu quả của thuật toán một cách chi tiết hơn.

TÀI LIỆU THAM KHẢO

[1] Quy Nguyen, Thuật toán phân cụm K-Means, https://ndquy.github.io/posts/thuat-toan-phan-cum-kmeans/, Ngày truy cập: 13/6/2024.

[2] CafeDev, Tự học ML Thuật toán phân cụm phân cụm K-Means Mini Batch, https://cafedev.vn/tu-hoc-ml-thuat-toan-phan-cum-k-mean-mini-batch/, Ngày truy cập: 13/6/2024.

[3] Hình ảnh

- https://laodong.vn/photo/net-dep-me-hoac-cua-nguoi-viet-qua-con-mat-cua-nhiep-anh-gia-nguoi-phap-840768.ldo, Ngày truy cập: 14/6/2024.
- https://wallpapers4screen.com/landscapes/vermillion-lake-4k-night-stars-banff-national-park-34264, Ngày truy cập: 14/6/2024.
- https://www.pixel4k.com/spring-autumn-colorful-nature-magical-forest-4k-85815.html, Ngày truy cập: 14/6/2024.