Đại học Quốc gia Thành phố Hồ Chí Minh

Đại học Khoa học Tự nhiên

Khoa Công Nghệ Thông Tin

Báo cáo đồ án

Lab: Project 1

Môn học: Toán ứng dụng và thống kê

Người thực hiện:
22127390 - Nguyễn Văn Lê Bá Thành - 22CLC08

Giảng viên:
Cô Phan Thị Phương Uyên
Thầy Nguyễn Ngọc Toàn
Thầy Nguyễn Văn Quang Huy
Thầy Vũ Quốc Hoàng

Ngày 18 tháng 6 năm 2024

Mục lục

1 Lời cảm ơn

Nếu không có sự giúp đỡ và hướng dẫn của nhiều người và từ nhiều nguồn khác nhau, đồ án này sẽ không thể hoàn thành.

Đầu tiên, em muốn cảm ơn các nhà phát triển và cộng tác viên của thư viện Numpy, Pillow và Matplotlib. Tài liệu có sẵn trên trang web của họ rất hữu ích trong việc làm quen và áp dụng để cài đặt các hàm cần thiết cho đồ án này

Em cũng muốn cảm ơn các thầy, cô hướng dẫn vì những lời khuyên hữu ích và những bài giảng với các kiến thức cần thiết trong suốt môn học này. Sự hiểu biết của em về dự án này đã được nâng cao hơn nhiều nhờ những lời giải thích rõ ràng và sự cởi mở với các câu hỏi.

Em đặc biệt muốn cảm ơn Gemini một mô hình ngôn ngữ lớn, vì đã giúp thu thập thông tin và tóm tắt thông tin về việc thực hiện color compression trên hình ảnh và cách hoạt động của thuật toán K-means.

2 Giới thiệu

Trong xã hội hiện tại hình ảnh kỹ thuật số rất cần thiết trong cuộc sống, từ việc lưu trữ kỷ niệm, nghiên cứu, nghệ thuật. Tuy nhiên, các tệp hình ảnh có độ phân giải cao thường có kích thước lớn và điều này sẽ dẫn đến việc xử lý, truyền tải và lưu trữ kém hiệu quả về mặt thời gian và không giam lưu trữ. Đó cũng là mục tiêu của kỹ thuật nén ảnh nhằm làm giảm kích thước ảnh mà không làm giảm chất lượng hình ảnh.

Báo cáo này thảo luận về việc áp dụng phương pháp nén màu K-mean nhằm làm giảm không gian lưu trữ cần thiết cho một hình ảnh. K-mean là một phương pháp học máy không giám sát, phân cụm các điểm dữ liệu dựa trên sự tương đồng. Kỹ thuật K-mean có thể làm giảm số lượng màu hiển thị trong hình ảnh khi nén hình ảnh và phương pháp này cung cấp sự cần bằng giữa độ trung thực của hình ảnh và tỷ lệ nén.

Các phần sau được tổ chức như sau, Phần 3 với các ý tưởng và mô tả về thuật toán K-mean và các hàm tiện ích. Phần 3 đưa ra kết quả với số lượng màu sắc cụ thể và nhận xét về các kết quả này. Các kết luận sẽ được đưa ra ở Phần 4.

3 Ý tưởng thực hiện và mô tả các hàm

3.1 Thuật toán K-means

3.1.1 Tổng quan về K-means

- **K-means là gì?**

  K-means là thuật toán không giám sát nổi tiếng dùng để phần vùng bộ tập dữ liệu thành k phân cụm cho trước. Mục tiêu của thuật toán này bao gồm: [1]

  1. Đưa các điểm dữ liệu trong bộ dữ liệu vào các phân cụm
  2. Các điểm dữ liệu trong cùng một phân cụm càng giống nhau càng tốt
  3. Các điểm phân cụm khác nhau càng khác nhau càng tốt

  Trong đồ án này, chúng ta sẽ sử dụng thuật toán K-means để cài đặt một chương trình có khả năng nén màu của một hình ảnh thành k màu cho trước.

- **Điều kiện dừng và sự hội tụ** Các tâm cụm hội tụ khi các giá trị tâm cụm mới không thay đổi sau các lần lặp, do đó điều kiện dừng của thuật toán K-means diễn ra dưới các trường hợp sau đây: [1]

  1. Các tâm cụm mới không thay đổi
  2. Các điểm dữ liệu nằm trong cùng một phân cụm
  3. Dat đến số lần lặp tối đa

3.1.2 Sử dụng K-means trong nén màu hình ảnh

- **Cấu tạo của ảnh:**

  Hình ảnh kỹ thuật số hiện nay được cấu tạo từ các điểm ảnh (pixel) có kích thước height x width với mỗi điểm ảnh có thể được biểu diễn dưới nhiều dạng khác nhau như HEX, RGB, ... [2]

  Trong đồ án này chúng ta sẽ sử dụng hệ màu RGB, với ba thuộc tính Red, Green, Blue là các số nguyên dương 8 bit bắt đầu từ 0 đến 255. Kết hợp ba thuộc tính ở trên ta có thể tạo ra một điểm ảnh có màu nằm trong $256 * 256 * 256 \approx 1.7 \times 10^7$ màu. [2]

  Dựa vào các ý trên ta có thể phân tích một hình ảnh thành các thành phần nhỏ như sau:

  1. Một điểm ảnh là một điểm dữ liệu
  2. Một điểm ảnh chứa 3 thuộc tính (properties) bao gồm 3 channels Red, Green, Blue
  3. Chiều dài (height) và độ rộng (width) của hình ảnh

  Vậy ta có thể xác định rằng một hình ảnh là một ma trận 3 chiều height, width, channels

- **Áp dụng K-means vào thuật toán nén ảnh**

  Ý tưởng của thuật toán nén được cài đặt trong đồ án này bao gồm:

  1. Làm phẳng ma trận điểm ảnh 3 chiều ban đầu height, width, channels trở thành ma trận 2 chiều height * width, channels.
  2. Khởi tạo giá trị tâm cụm (centroids) ban đầu bằng cách chọn ngẫu nhiên hoặc chọn từ các giá trị RGB của ảnh và khơi tạo một mảng labels nguyên có kích thước height * width dùng để chỉ định pixel nào ứng với tâm cụm nào.
  3. Tính toán khoảng cách giữa các điểm ảnh và các phân cụm sau đó đưa điểm ảnh vào phân cụm có khoảng cách ngắn nhất so với điểm ảnh đó bằng cách gán nhãn.
  4. Tính toán lại giá trị tâm cụm (centroids) bằng cách tính trung bình các giá trị được phân loại vào phân cụm tương ứng.
  5. Sử dụng các giá trị tâm cụm mới này và lặp lại bước 3 cho đến khi đạt đến số vòng lặp tối đa hoặc giá trị tâm cụm mới giống với giá trị tâm cụm cũ (hội tụ).
  6. Tao ra một ma trận 2 chiều height * width, channels bằng cách gán giá trị tâm cụm ứng với giá trị trong mảng labels và do labels có kích thước height * width, do đó ma trận 2 chiều được tạo mới sẽ có height và width giống với ảnh gốc và chứa các giá trị tâm cụm.
  7. Reshape ma trận 2 chiều heigh * width, channels này thành ma trận 3 chiều height, width, channels.

  Kết quả trả về là một hình ảnh có kích thước giống kích thước ảnh ban đầu với k màu được nén lại

3.2 Mô tả các hàn trong chương trình

3.2.1 Các thư viện cần dùng

- **Các thư viện bắt buộc sử dụng bao gồm:**
  1. Numpy: dùng để tính toán ma trận
  2. PIL: dùng để đọc, ghi ảnh
  3. matplotlib: dùng để hiển thị ảnh
- **Các thư viện được sử dụng để test và viết báo cáo**
  1. Time: dùng để tính toán thời gian thực thi
  2. BytesIO: dùng để tính kích thước của file hình ảnh

3.2.2 Các hàm trong chương trình

1. **def read_img(img_path):**
   - **Mô tả hàm:** read_img được sử dụng để đọc hình ảnh từ đường dẫn cho trước.
   - **Tham số đầu vào:** nhận một biến string chứa đường dẫn đến hình ảnh cần nén
   - **Các bước thực hiện:** sử dụng hàm open và hàm convert trong thư viện Pillow với hàm convert sử dụng chế độ RGB để chuyển đổi hình ảnh được đọc thành hệ màu RGB [3].
   - **Kết quả đầu ra:** mọt object dạng "PIL.Image.Image"
2. **def show_img(img_2d):**
   - **Mô tả hàm:** show_img giúp đưa hình ảnh ra ngoài màn hình.
   - **Tham số đầu vào:** nhận biến img_2d là một object có kiểu dữ liệu "ArrayLike" hoặc "PIL.Image.Image". Trong đó kiểu dữ liệu "ArrayLike" là các object có thể được chuyển đổi thành một mảng [5].
   - **Các bước thực hiện:** dùng method imshow của thư viện matplotlib sẽ xử lý ảnh được truyền vào [4].
   - **Kết quả đầu ra:** hình ảnh được thể hiện trên màn hình

3. **def save_img(img_2d, img_path):**

   - **Mô tả hàm:** save_img là hàm giúp lưu hình ảnh với đường dẫn cho sẵn.
   - **Tham số đầu vào:** hàm nhận hai biến số đầu vào img_2d là ma trận điểm ảnh ndarray và img_path là đường dẫn đến tập tin đích.
   - **Các bước thực hiện:** ta sử dụng hàm save trong thư viện numpy và truyền tham số img_path vào [5].
   - **Kết quả đầu ra:** kết quả trả ra của hàm này là một file hình ảnh được lưu ở đường dẫn đích

   *Lưu ý: vì hàm save_img được sử dụng sau khi quá trình xử lý nén ảnh đã hoàn tất, và ma trận được tạo mới từ các tâm cụm và các nhãn sẽ có kiểu dữ liệu ndarray.

4. **def convert_img_to_1d(img_2d):**

   - **Mô tả hàm:** conver_img_to_1d là hàm giúp làm phẳng một ma trận bằng cách giảm đi một chiều của ma trận thông qua hàm reshape của thư viện numpy [6].
   - **Tham số đầu vào:** hàm nhận một đối số đầu vào là img_2d, trong trường hợp này chính là ma trận được trả về từ hàm read_img vì vậy img_2d có kiểu dữ liệu là "PIL.Image.Image".
   - **Các bước thực hiện:** ta lấy chiều dài và chiều rộng của ma trận thông qua method size được định nghĩa sẵn trong thư viện PIL [3]. Sau đó, ta gọi method reshape của thư viện numpy để reshape lại ma trận và method astype(int) để đảm bảo các phần tử trong ma trận là số nguyên.
   - **Kết quả đầu ra:** một ma trận cấp hai có kích thước (height * width) x 3

5. **def init_centroid_value(img_1d, k_clusters, init_centroids):**

   - **Mô tả hàm:** init_centroid_value là hàm dùng để khởi tạo giá trị tâm cụm ban đầu bằng cách chọn random hoặc chọn một điểm ảnh trong hình.
   - **Tham số đầu vào:** hàm nhận ba tham số đầu vào bao gồm img_1d là ma trận đã được làm phẳng, k_clusters là số màu để gom cụm và init_centroids là cách chọn tâm cụm.
   - **Các bước thực hiện:**

     B1: Tạo ra một ma trận mới với các dòng không trùng nhau dựa trên ma trận img_1d bằng method unique trong thư viện numpy với axis = 0 để chọn ra các điểm ảnh duy nhất dựa trên số dòng của ma trận img_1d.

     B2: Dựa vào biến init_centroids để chọn tâm cụm.

     Nếu init_centroids là random ta sẽ tạo ra một ma trận centroids bằng method np.random.randint để chọn ngẫu nhiên các số từ 0 đến 255 cho từng phần tử trong ma trận centroids với kích thước k_clusters x channels

     Nếu init_centroids là in_pixels ta sẽ tạo ra một ma trận centroids bằng method np.random.choice để chọn ngẫu nhiên các dòng trong ma trận img_1d cho từng dòng trong ma trận centroids với kích thước k_clusters x channels.

   - **Kết quả đầu ra:** Một ma trận centroids có kích thước k_clusters x channels.

6. **def kmeans(img_1d, k_clusters, max_iter, init_centroids):**

   - **Mô tả hàm:** keams là hàm được dùng để cài đặt thuật toán K-means để tìm các tâm cụm và gán nhãn cho các điểm ảnh trong hình.
   - **Tham số đầu vào:** hàm nhận bốn đối số đầu vào bao gồm: img_1d là ma trận đã được làm phẳng, k_clusters là số màu để gom cụm, max_iter là số vòng lặp tối đa và init_centroids là cách chọn tâm cụm
   - **Các bước thực hiện:**

     B1: Sử dụng hàm init_centroid_value và truyền các đối số img_1d, k_clusters, init_centroids vào để chọn ra các tâm cụm ban đầu

     B2: Sao chép mảng centroids sang một array khác là centroids_old để lưu lại giá trị tâm cụm cũ.

     B3: Tính toán khoảng cách giữa các điểm ảnh và các tâm cụm. Trong đồ án này, chúng ta sẽ sử dụng np.linalg.norm để tính khoảng cách euclid giữa các điểm ảnh và tâm cụm với axis = 2 để thực hiện phép toán trên chiều cuối cùng của ma trận đó là chiều của channel [7].

     Để tính khoảng cách euclid giữa các tâm cụm với từng điểm ảnh. Đầu tiên ta cần hiệu giữa các tâm cụm k với từng điểm ảnh bằng cách sử dụng khả năng broadcast trong thư viện numpy. Broadcasting là một cơ chế mạnh mẽ cho phép thực thi các

phép toán số học trên các numpy array có kích thước khác nhau. Chúng ta thường có một mảng nhỏ hơn và một mảng lớn hơn và chúng tôi muốn sử dụng mảng nhỏ hơn nhiều lần để thực hiện một số thao tác trên mảng lớn hơn [8]. Khi thực hiện broadcasting trên hai mảng, Numpy sẽ so sánh số chiều của hai mảng bắt đầu từ trong ra ngoài và hai ma trận tương thích khi

- Cả hai ma trận có chiều giống nhau, hoặc
- Một trong hai ma trận có một chiều là 1

Như đã biết, img_1d là một ma trận hai chiều có kích thước height * width, channels và centroids là ma trận có kích thước k_cluster, channels nên hai ma trận này có cùng chiều do $(height*width) \neq k\_cluster$ và để có thể thực hiện được broadcasting ta cần đưa một trong hai ma trận về dạng tương thích. Trong trường hợp này, chúng ta sẽ chọn cách chèn thêm một chiều vào ma trận centroids bằng cách sử dụng np.newaxis. Để thực hiện chèn thêm một chiều ta sử dụng lệnh centroids[:, np.newaxis] để tách channels ra thành cột mới và số chiều của ma trận centroids sẽ trở thành k_cluster, 1, channel [9].

Vậy khi việc chèn thêm chiều kết thúc ta có hai ma trận centroids mới có dạt điều kiện để thực hiện broadcasting (điều kiện thứ hai với dòng của ma trận centroids bây giờ là 1). Trong quá trình broadcasting từng điểm ảnh của img_1d sẽ được trừ cho từng dòng của ma trận centroids và trả ra một ma trận kết quả có kích thước k_cluster, height * width, channel.

Ma trân kết quả này sẽ được sử dụng để tính toán khoảng cách euclid, và do np.linalg.norm sử dụng axis=2 để tính toán nên ma trận distance sẽ có kích thước là k_cluster, height * width.

B4: Tiếp đến ta sử dụng hàm np.argmin với axis = 0 để gán nhãn cho các điểm ảnh bằng cách tìm khoảng cách nhỏ nhất trong ma trận distance. Do hàm np.argmin chạy với axis = 0, nên argmin sẽ tìm dựa theo dòng (nghĩa là tìm khoảng cách nhỏ nhất trong ba dòng cùng cột). Kết quả trả ra sẽ là vị trí của dòng trong ma trận distance có khoảng cách euclid nhỏ nhất và vì ma trận distance có height * width cột nên ma trận labels sẽ là một ma trận một chiều kích thước height * width chứa nhãn của điểm ảnh tương ứng trong img_1d.

B5: Tiếp theo ta cần tính toán lại tâm cụm bằng cách tính trung bình của các điểm ảnh dựa vào ma trận labels. Để tìm được các điểm ảnh được phân vào cùng một cụm, ta dùng một vòng lặp bắt đầu từ i = 0 đến số màu cần gom cụm (k_cluster). và sử dụng lệnh img_1d[labels == i] để truy cập vào các điểm ảnh trong cùng một phân cụm, sau đó sử dụng hàm len để kiểm tra xem các điểm ảnh đó có chứa rỗng hay không. Nếu có chứa rỗng rỗng ta sẽ khởi tạo một ma trận 0 bằng np.zeros với kích thước 1, channels và gán ma trận này vào centroids thứ i. Nếu không chứa rỗng ta dùng np.mean với axis = 0 để tính trung bình giữa các dòng trong cùng một phân cụm sau đó gán vào centroids thứ i.

B6: Sau khi có được ma trận centroids mới ta sử dụng hàm array_equal trong thư viện numpy để so sánh hai ma trận centroids mới toạ và centroids_old. Nếu trả ra kết quả đúng, ta kết thúc thuật toán k-means do các tâm cụm đã hội tụ. Nếu kết quả là không đúng, ta tiếp tục quay lại B2.

- **Kết quả đầu ra:** Hai ma trận bao gồm ma trận centroids chứa các tâm cụm đã hội tự và ma trận labels chứa nhãn của từng điểm ảnh.

7. **def generate_2d_img(img_2d_shape, centroids, labels):**

- **Mô tả hàm:** generate_2d_img giúp tạo mới một ảnh 2d dựa trên các giá trị trả ra của thuật toán K-means.
- **Tham số đầu vào:** hàm nhận vào ba đối số bao gồm: img_2d_shape chứa kích thước của ảnh gốc, centroids các giá trị tâm cụm, labels ma trận nhãn ứng với từng điểm ảnh.
- **Các bước thực hiện:**

(a) Ta lấy các giả trị tâm cụm và gán chúng vào ma trận result dựa vào ma trận labels. Điều này có thể được thực hiện bằng câu lệnh sau result = centroids[labels].

(b) Sau đó do, mảng label là mảng 1 chiều nên ta cần phải reshape ma trận result lại để có thể tạo ra một ma trận điểm ảnh. Điều này được thực hiện bằng cách kết hợp reshape vs câu lệnh trên result = centroids[labels].reshape(img_2d_shape[0], img_2d_shape[1], 3) với img_2d_shape[0] là chiều cao và img_2d_shape[1] là chiều rộng của ảnh.

(c) Sau đó để giúp việc lưu trữ và xuất hình ảnh dễ dàng hơn em sử dụng hàm np.unint8 để chuyển đổi các giá trị trong ma trận result thành các số nguyên 8 bit không âm và hàm Image.fromarray() trong thư viện PIL để tạo ra một object từ một ma trận.

- **Kết quả đầu ra:** Một object "PIL.Image.Image" với các điểm ảnh là các tâm cụm

8. **def main():**

- **Mô tả hàm:** main là xử lý input và gọi các hàm để thực thi các chức năng.
- **Các bước thực hiện:**

B1: Cho phép người dùng nhập vào đường dẫn của ảnh, nhập số vòng lặp tối đa, số lượng màu cần nén, loại tâm cụm ban đầu và định dạng mà ảnh sẽ được lưu.

B2: Dựa vào lựa chọn loại centroid ta sẽ gán các giá trị tương ứng cho chúng

- $centroidT = 0 \rightarrow random$
- $centroidT = 1 \rightarrow in\_pixels$

B3: Dựa vào lựa chọn định dạng ảnh ta sẽ gán các giá trị tương ứng cho chúng

- $data\_type = 0 \rightarrow png$
- $data\_type = 1 \rightarrow pdf$
- $data\_type = 2 \rightarrow jpg$

B4: Tiếp theo ta tiền hành phân tách img_path để lấy tên ảnh và đường dẫn đến folder đang chưa ảnh bằng các kỹ thuật indexing và hàm split sau đó lưu lại để sử dụng sau.

B5: Ta sử dụng hàm read_img và truyền vào img_path để lấy được ma trận điểm ảnh gốc.

B6: Do object trả về từ hàm read_img có kiểu "PIL.Image.Image" không có thuộc tính shape nên ta chuyển đổi object này thành ndarray để có thể sử dụng thuôc tính shape nhằm lấy ra các thông số về chiều cao, chiều rộng và channel.

B7: Sau đó ta tiến hành làm phẳng ma trận điểm ảnh bằng hàm conver_img_to_1d với đối số đầu vào là object trả về từ hàm read_img.

B8: Ta tiếp tục gọi hàm kmeans và truyền các tham số đầu vào bao gồm: ma trận đã được làm phẳng, số lượng ảnh cần nền, số vòng lặp tối đa và loại tâm cụm ban đầu. Biến centroids và labels sẽ chứa kết quả trả ra của hàm này.

B9: Để tạo ra một hình ảnh hoàn chỉnh ta sử dụng hàm generate_2d_img với các đối số được truyền vào bao gồm: shape của ma trận gốc. centroids và labels. Với kết quả trả về một object có kiểu "PIL.Image.Image".

B10: Sau ta sử dụng đường dẫn và tên file đã được phân tách từ trước để tạo nên một tên mới kết hợp với định dạng ảnh được chọn, ta sẽ có một bức ảnh được lưu cùng thư mục với ảnh gốc và có tên là "ảnh gốc_số lượng màu nén_compressed.định dạng đã chọn".

B11: Cuối cùng ta in hình ảnh được trả ra với hàm show_img.

4 Kết quả kiểm thử

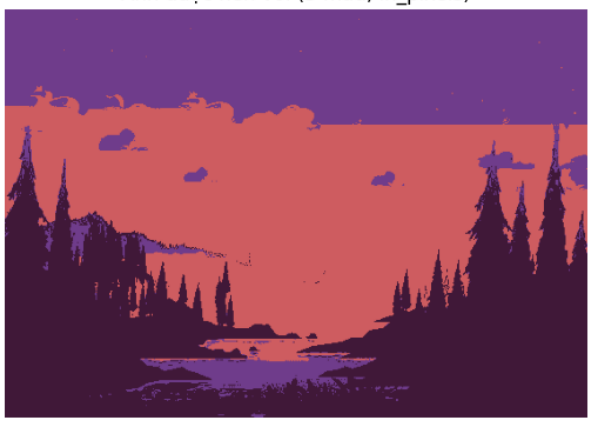
Các mẫu thử này được chạy trên cấu hình như sau: CPU: Intel I5-9600K, Ram: 16GB x 4 Gskill Trident Z, Mainboard: Aorus B360 pro. Sử dụng kernel python 3.11.9 để xử lý trên Jupyter notebook, các mẫu thử này là kết quả trả ra của hàm test, thầy/cô có thể mở comment code trong phần your test để chạy thử.

4.1 Mẫu thử 1: ảnh độ phân giải thấp (593 x 394)

- **Ảnh gốc**

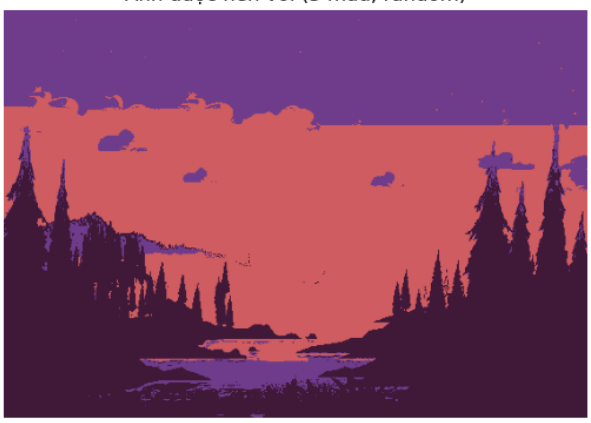
- **Ảnh do K-means được cài đặt nén**

Ảnh được nén với (3 màu, in_pixels)



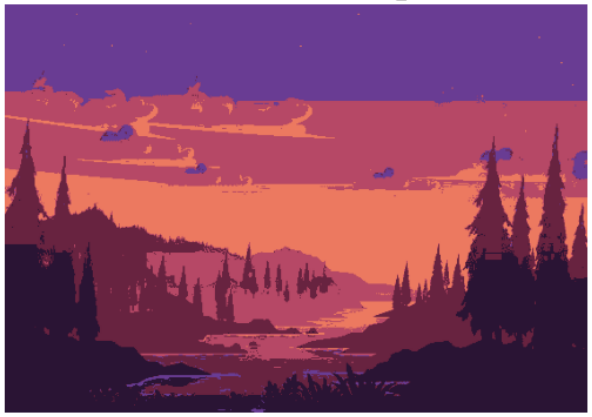
Thời gian chạy 0.563s.
Dung lượng: 16.626 KB
Giảm 33.633% dung lượng

Ảnh được nén với (3 màu, random)



Thời gian chạy 0.542s.
Dung lượng: 16.605 KB
Giảm 33.715% dung lượng

Ảnh được nén với (5 màu, in_pixels)



Thời gian chạy 1.896s.
Dung lượng: 20.372 KB
Giảm 18.680% dung lượng

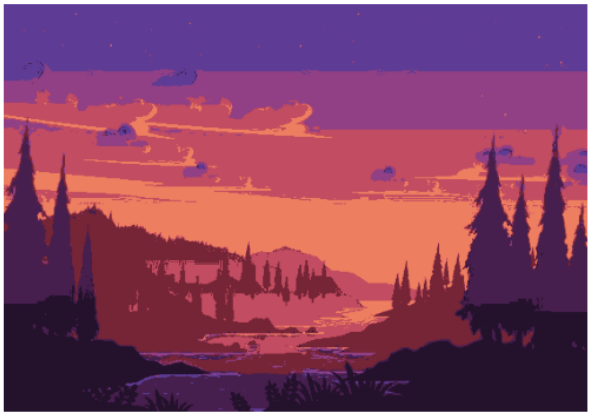
Ảnh được nén với (5 màu, random)

Thời gian chạy 1.053s.
Dung lượng: 20.477 KB
Giảm 18.263% dung lượng

Ảnh được nén với (7 màu, in_pixels)

Thời gian chạy 2.109s.
Dung lượng: 20.485 KB
Giảm 18.228% dung lượng

Ảnh được nén với (7 màu, random)



Thời gian chạy 1.773s.
Dung lượng: 20.868 KB
Giảm 16.700% dung lượng

4.2 Mẫu thử 2: ảnh độ phân giải trung bình (1600 x 800)

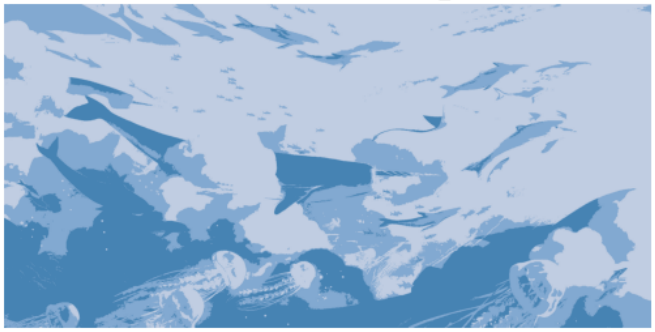
- **Ảnh gốc**

Hình ảnh gốc (91257 màu)

Dung lượng: 128.085 KB

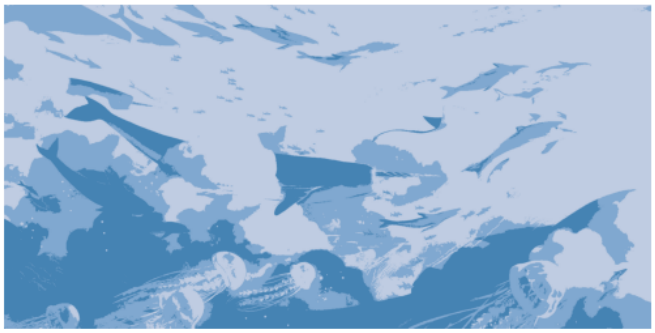
- **Ảnh do K-means được cài đặt nén**

Ảnh được nén với (3 màu, in_pixels)



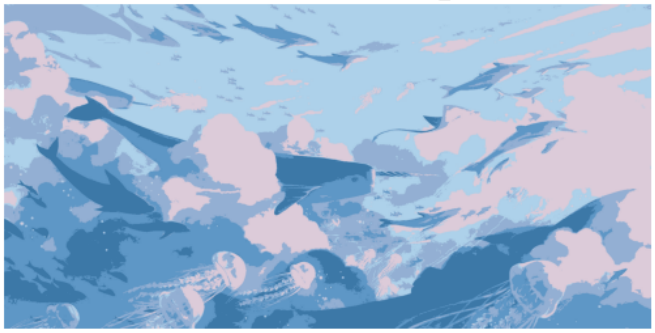
Thời gian chạy 6.073s.
Dung lượng: 97.758 KB
Giảm 23.677% dung lượng

Ảnh được nén với (3 màu, random)



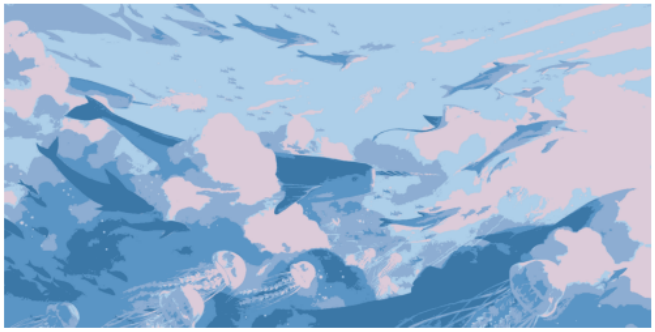
Thời gian chạy 7.179s.
Dung lượng: 96.383 KB
Giảm 24.751% dung lượng

Ảnh được nén với (5 màu, in_pixels)



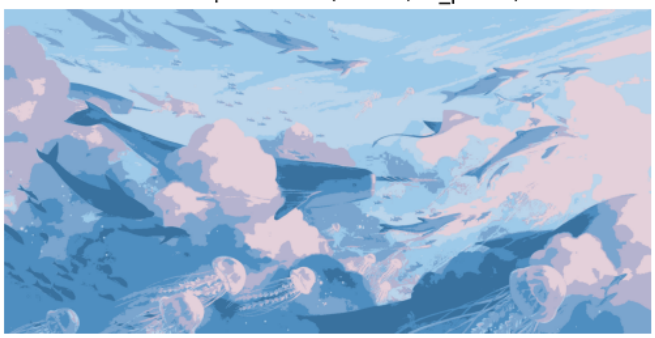
Thời gian chạy 15.199s.
Dung lượng: 107.303 KB
Giảm 16.225% dung lượng

Ảnh được nén với (5 màu, random)



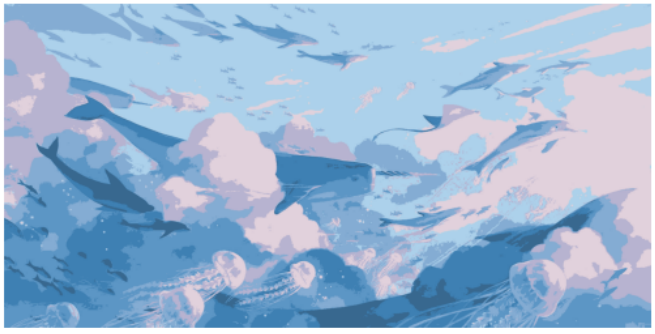
Thời gian chạy 19.253s.
Dung lượng: 106.388 KB
Giảm 16.940% dung lượng

Ảnh được nén với (7 màu, in_pixels)



Thời gian chạy 6.450s.
Dung lượng: 115.852 KB
Giảm 9.551% dung lượng

Ảnh được nén với (7 màu, random)

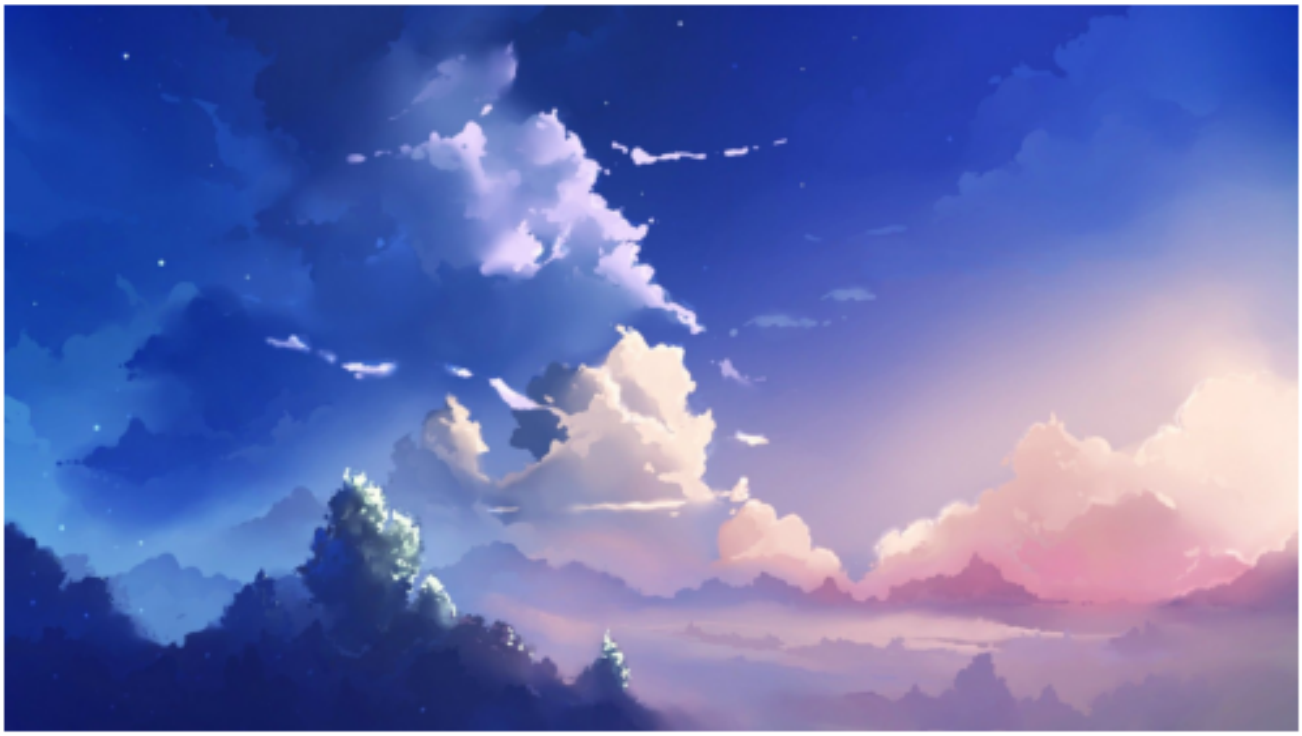


Thời gian chạy 17.239s.
Dung lượng: 112.124 KB
Giảm 12.461% dung lượng

4.3 Mẫu thử 3: ảnh độ phân giải cao (1920 x 1080)

- **Ảnh gốc**

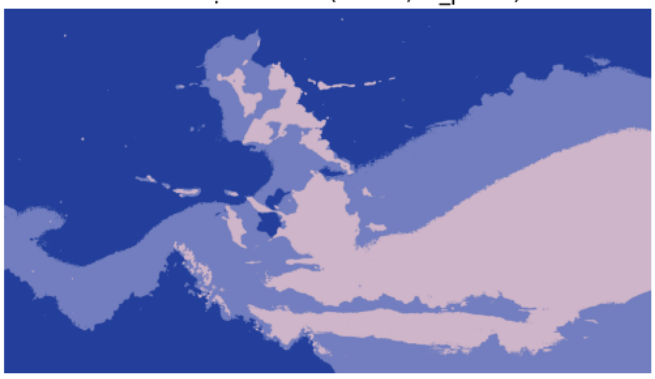
Hình ảnh gốc (99128 màu)



Dung lượng: 121.318 KB

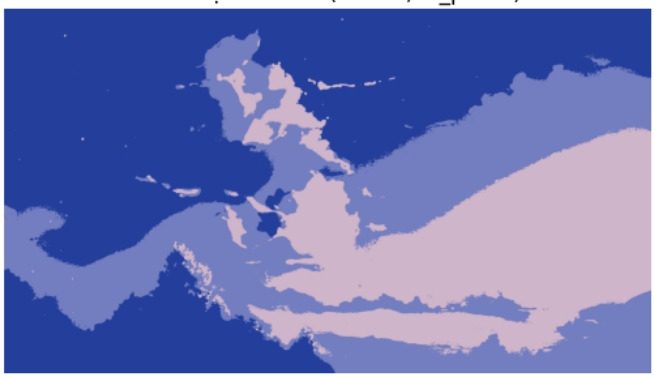
- **Ảnh do K-means được cài đặt nén**

Ảnh được nén với (3 màu, in_pixels)



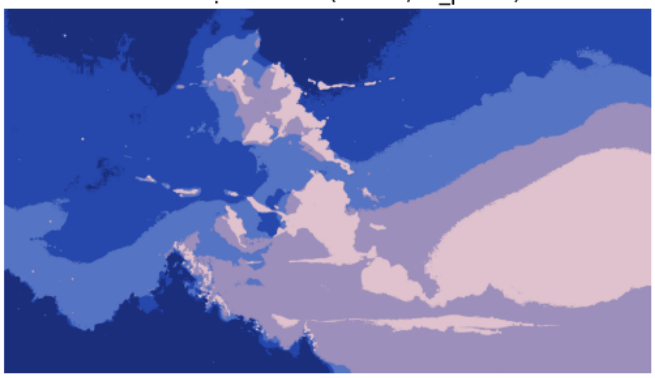
Thời gian chạy 14.845s.
Dung lượng: 90.972 KB
Giảm 25.014% dung lượng

Ảnh được nén với (3 màu, in_pixels)



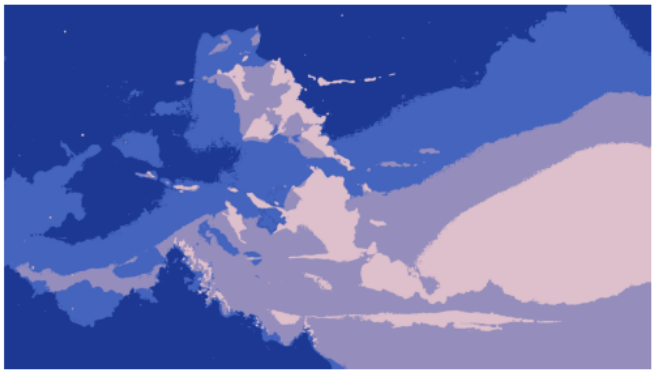
Thời gian chạy 14.845s.
Dung lượng: 90.972 KB
Giảm 25.014% dung lượng

Ảnh được nén với (5 màu, in_pixels)



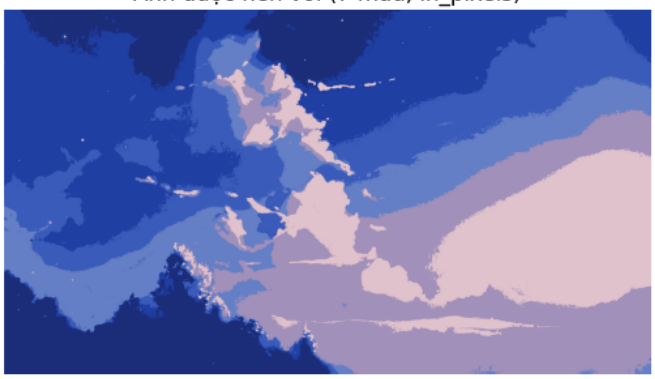
Thời gian chạy 23.892s.
Dung lượng: 97.040 KB
Giảm 20.012% dung lượng

Ảnh được nén với (5 màu, random)



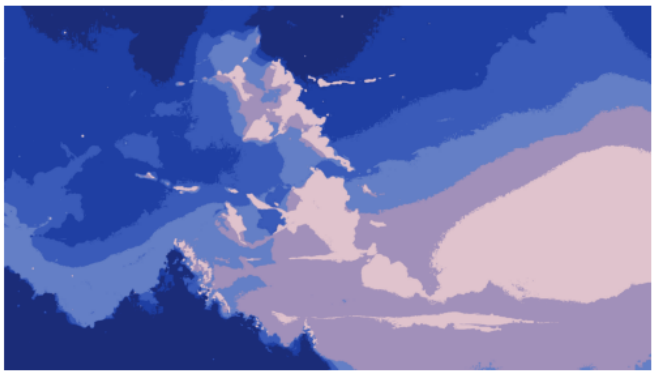
Thời gian chạy 12.669s.
Dung lượng: 98.423 KB
Giảm 18.872% dung lượng

Ảnh được nén với (7 màu, in_pixels)



Thời gian chạy 49.233s.
Dung lượng: 100.065 KB
Giảm 17.518% dung lượng

Ảnh được nén với (7 màu, random)



Thời gian chạy 15.841s.
Dung lượng: 100.304 KB
Giảm 17.322% dung lượng

5 Nhận xét

Trong phần này em sẽ đưa ra các nhân xét dựa trên các kết quả trả ra bao gồm thời gian hoàn thành, dung lượng sau khi nén và độ chính xác khi so sánh với thuật toán k-means của thư viện Scikit-learn. Ngoài ra em cũng sẽ đánh giá và xác định các yếu tố ảnh hưởng đến hiệu năng và kết quả của thuật toán K-means.

Trên cả ba bộ dữ liệu được chạy thử, chúng ta có thể thấy được với k tăng dần dung lượng của ảnh ngày càng tăng và càng tiến gần đến dung lượng của ảnh gốc, vì với k càng lớn số lương màu càng nhiều dẫn đến dung lượng tăng tương ứng. Vậy ta có thể kết luận rằng việc chọn k có thể dẫn đến sự tăng hoặc giảm của dung lượng ảnh.

Tuy nhiên sự tăng hoặc giảm k chưa chắc dẫn đến sự tăng hoặc giảm của thời gian chạy, dựa vào các bộ dữ liệu ở trên ta có thể thấy được ở mẫu thử 2 thời gian thực thi ở k = 7 nhỏ hơn thời gian thực thi của k = 5. Điều này có thể được giải thích bằng sự ngẫu nhiên trong cách chọn tâm cụm ban đầu, với các tâm cụm khác nhau chúng ta có thể có các kết quả khác nhau do đó cũng có những tâm cụm hội tụ nhanh hơn [10, 11].

Số lượng màu sắc trong ảnh cũng gây ảnh hưởng hiệu năng của thuật toán K-means. Trong các hình ảnh có nhiều màu sắc hoặc sử dụng màu gradient, với màu gradient là sự dịch chuyển màu giữa hai màu do đó tạo nên rất nhiều màu sắc khác nhau (ví dụ như các mẫu thử 1, 2 và 3 đều có màu gradient) [12]. Ta có thể thấy rằng với k = 3, 5, 7 ta không thể thể hiện đầy đủ các vùng có màu gradient, vì thể để có thể thể hiện đầy đủ một ảnh có màu gradient ta cần phải chọn k lớn hoặc rất lớn để có thể biểu diễn màu đầy đủ các màu và màu gradient. Tuy nhiên chọn k lớn cũng đồng nghĩa với việc thời gian xử lý sẽ tăng lên như đã nói ở trên, vì vậy ta có thể nói rằng K-means không phù hợp để xử lý các hình ảnh có màu gradient, hoặc chứa nhiều màu sắc.

Ngoài ra độ phân giải của ảnh cũng sẽ ảnh hướng đến thời gian thực thi của thuật toán k-means. Với mẫu thử 1, ta có thể thấy thời gian chạy rất nhanh do số lượng điểm ảnh có trên ảnh không nhiều. Còn đối voi những mẫu thử có độ phân giải lớn hơn như mẫu thử 2, 3. Ta có thể thấy thời gian thực thi trung bình cao hơn nhiều so với mẫu thử 1 do số lượng điểm ảnh cần được xử lý lớn hơn.

Thuật toán K-means là một thuật toán mất mát do các hình ảnh đươc nén dựa trên tâm cụm gần nhất vì thế thuật toán K-means gây mất mát dữ liệu trong quá trình này và do đó các hình ảnh đã nén không thể đảo ngược lại [11].

Vậy ta có thể kết luận những yếu tố ảnh hướng đến hiệu năng của thuật toán K-means bao gồm ba yếu tố như sau:

- Số lượng màu sắc, màu gradient
- Độ phân giải của ảnh
- Số lượng màu cần nén

6 Kết luận

Thuật toán K-means là một thuật toán đơn giản và khá dễ hiểu, và khi được tối ưu tốt nó có thể được sử dụng để nén ảnh trong thời gian thực và xử lý các ảnh cực lớn như thuật toán Kmeans của thư viện Scikit-learn.

Tuy nhiên, người dùng sẽ cần phải lưu ý về các yếu tố ảnh hướng đến hiệu năng của thuật toán ví dụ như sử dụng trên các ảnh có màu gradient, đánh đổi giữa chất lượng hình ảnh và số lượng màu nén, và sự ngẫu nhiên khi thực hiện khởi tao giá trị ban đầu cho tâm cụm.

Tài liệu

[1] P. Sharma, "The Most Comprehensive Guide to K-Means Clustering You'll Ever Need," Analytics Vidhya, 19/8/2019 https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/, Accessed date: 14/6/2024

[2] S. Hoekstra, "What is RGB Color?," Nix Sensor Ltd, 14/2/2018. https://www.nixsensor.com/blog/what-is-rgb-color/, Accessed date: 14/6/2024

[3] "Image Module — Pillow (PIL Fork) 6.2.1 documentation," Readthedocs.io, 2011. https://pillow.readthedocs.io/en/stable/reference/Image.html, Accessed date: 14/6/2024

[4] "matplotlib.pyplot — Matplotlib 3.5.2 documentation," matplotlib.org. https://matplotlib.org/stable/api/pyplot_summary.html, Accessed date: 14/6/2024

[5] "Input and output — NumPy v2.0 Manual," numpy.org. https://numpy.org/doc/stable/reference/routines.io.html, Accessed Date: 15/6/2024.

[6] "Array manipulation routines — NumPy v1.24 Manual," numpy.org. https://numpy.org/doc/stable/reference/routines.array-manipulation.html, Accessed date: 15/6/2024

[7] "numpy.linalg.norm — NumPy v1.20 Manual," numpy.org. https://numpy.org/doc/stable/reference/generated/numpy.linalg.norm.html, Accessed date: 16/6/2024

[8] "Broadcasting — NumPy v1.22 Manual," numpy.org. https://numpy.org/doc/stable/user/basics.broadcasting.html, Accessed date: 16/6/2024

[9] "Applied-Mathematics-and-Statistics/lab02_project01/20127641.ipynb at main · NgocTien0110/Applied-Mathematics-and-Statistics," GitHub. https://github.com/NgocTien0110/Applied-Mathematics-and-Statistics/blob/main/lab02_project01/20127641.ipynb, Accessed Date: 16/6/2024).

[10] M. Kumar Gupta and P. Chandra, "P-k-means: k-means using Partition based Cluster Initialization Method under responsibility of International Conference on Advancements in Computing & Management." Accessed date: 16/6/2024. [Online]. Available: https://events.rdias.ac.in/wp-content/uploads/2021/06/P-k-means.pdf

[11] S. Charmot, "Clear, Visual Explanation of K-Means for Image Compression," Medium, Feb. 19, 2023. https://medium.com/@sebastian.charmot, Accessed date: 17/6/2024.

[12] "Color gradients: Definition and types | Freepik Blog," www.freepik.com, 15/5/2024. https://www.freepik.com/blog/color-gradient/#What_is_a_color_gradient, Accessed date: 16/6/2024).