

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

---o0o---



BÁO CÁO ĐỒ ÁN 1 COLOR COMPRESSION

Môn học: Toán ứng dụng và thống kê cho CNTT

Giảng viên hướng dẫn: Phan Thị Phương Uyên

Nguyễn Văn Quang Huy

Sinh viên thực hiện: Đoàn Thị Yến Nhi

MSSV: 21127660

Thành phố Hồ Chí Minh, tháng 7 năm 2023

Nội dung

1. Tìm hiểu về thuật toán K-means	3
2. Ý tưởng thực hiện	3
3. Mô tả các hàm	6
4. Kết quả	11
5. Nhận xét kết quả	13
6. Tài liệu tham khảo.....	15

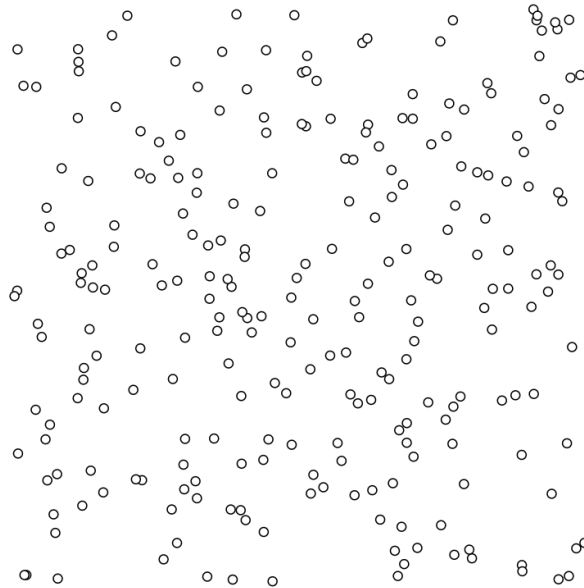
1. Tìm hiểu về thuật toán K-means

K-mean clustering là một phương pháp để tìm các cụm và hạt nhân – trung tâm của cụm trong mong muốn phân chẳng hạn như k cụm. Thuật toán K-mean di chuyển lặp đi lặp lại các hạt nhân để giảm thiểu tổng số trong phương sai cụm. Với một tập hợp các hạt nhân ban đầu, thuật toán K-means lặp lại hai bước:

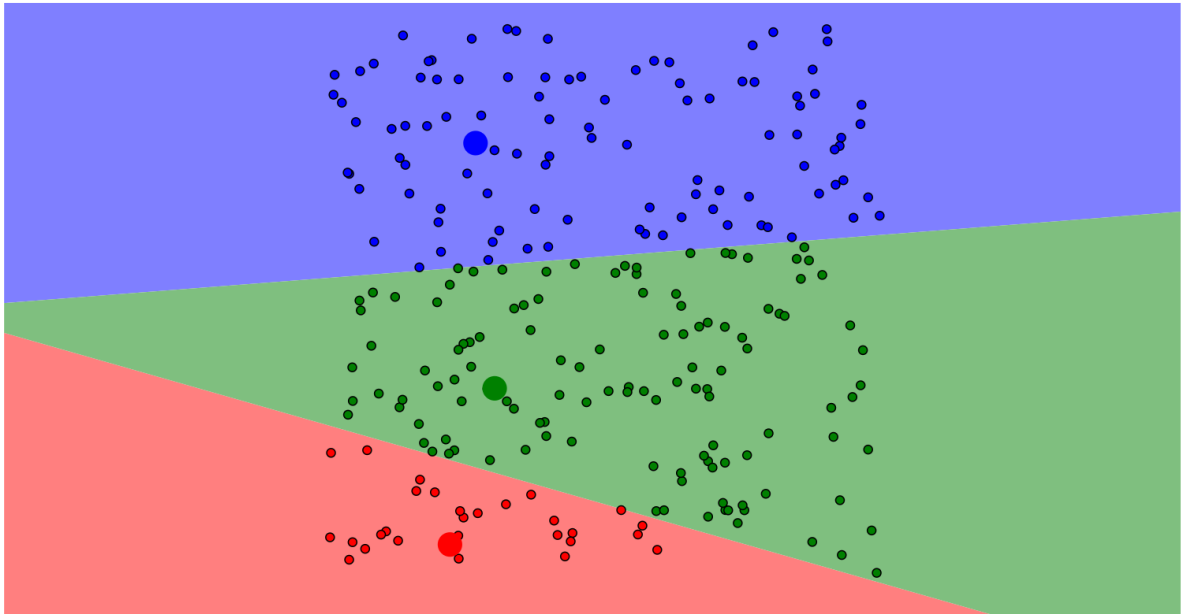
- Đối với mỗi hạt nhân, tính toán khoảng cách giữa các training point với nó và nếu gần nó hơn thì sẽ gán là cụm của hạt nhân đấy.
- Sau khi phân được cụm như ở bước trên, thì tiếp theo các training point của các cụm tính toán vector trung bình để được vị trí của hạt nhân mới và lặp lại bước trên đến khi không thể thay đổi được vị trí hạt nhân nữa.

2. Ý tưởng thực hiện

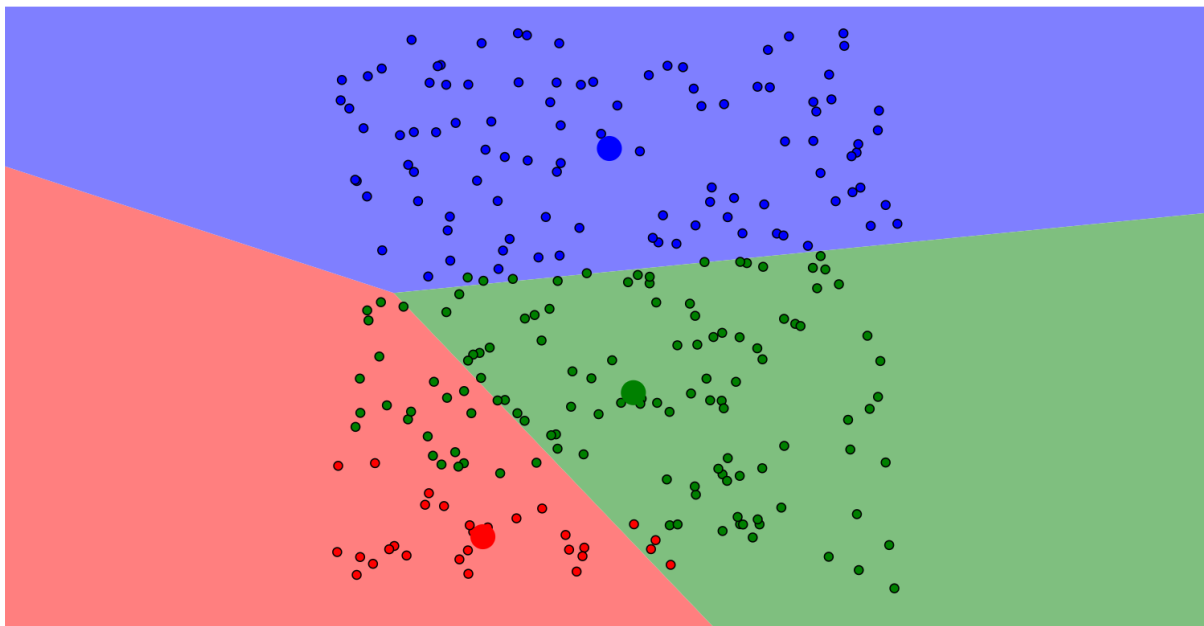
- **Hình 1:** Ban đầu ta có các điểm dữ liệu



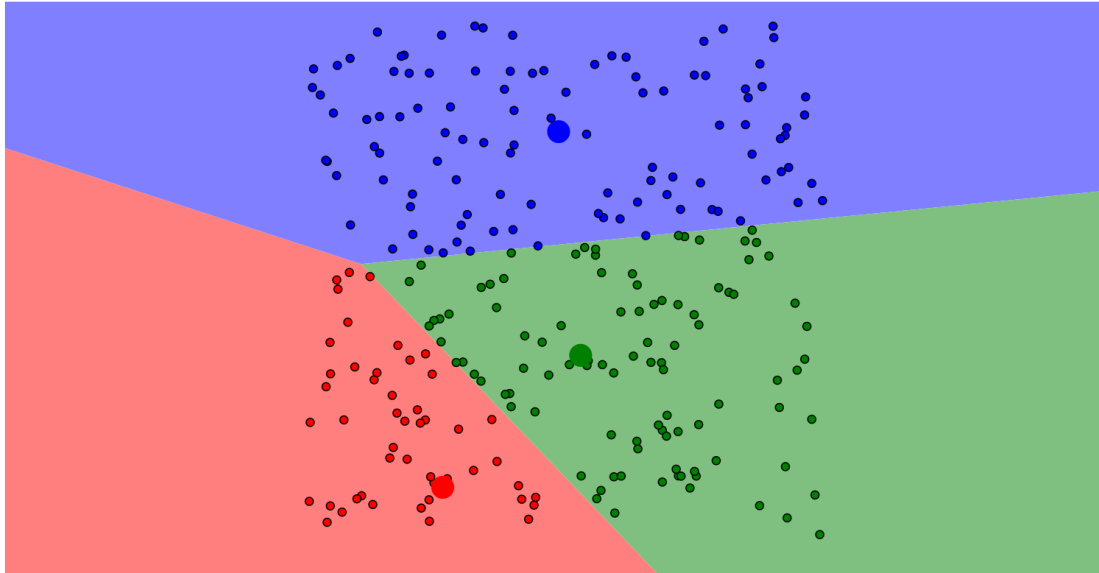
- **Hình 2:** Chọn số lượng nhóm K muốn phân loại dữ liệu. Chọn K điểm bất kỳ trong tập dữ liệu làm điểm trung tâm ban đầu cho từng nhóm. Gán từng điểm dữ liệu vào nhóm gần nhất (có khoảng cách gần nhất) với điểm trung tâm.



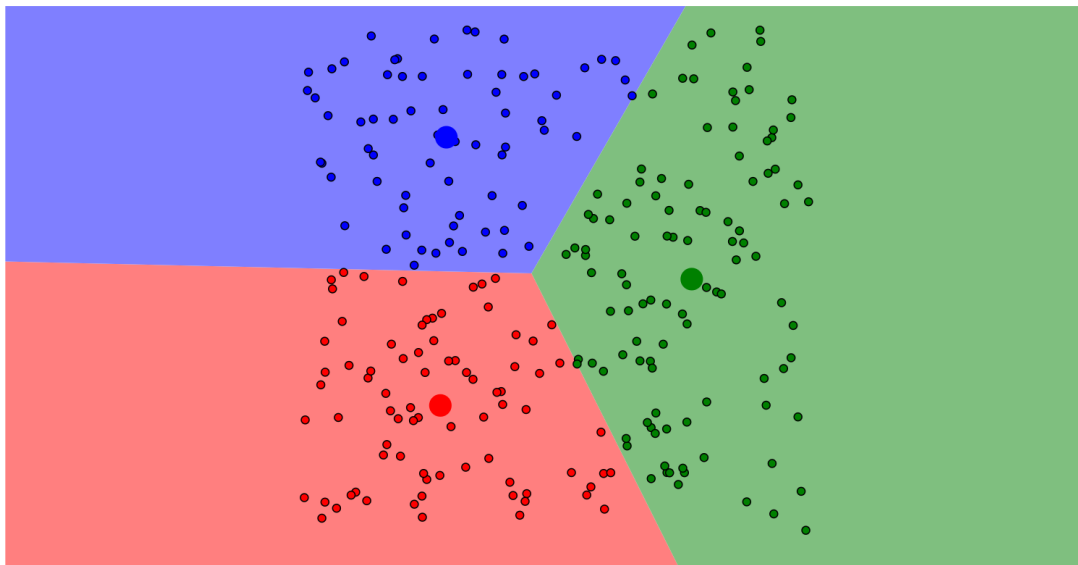
- **Hình 3:** Tính toán lại các điểm trung tâm mới của mỗi nhóm bằng cách lấy trung bình của các điểm dữ liệu trong nhóm đó.



- **Hình 4:** Gán lại điểm dữ liệu



- **Hình 5:** Lặp đi lặp lại các bước trên cho đến khi không còn sự thay đổi giữa các điểm trung tâm. Kết quả là các nhóm dữ liệu đã được phân loại thành công.



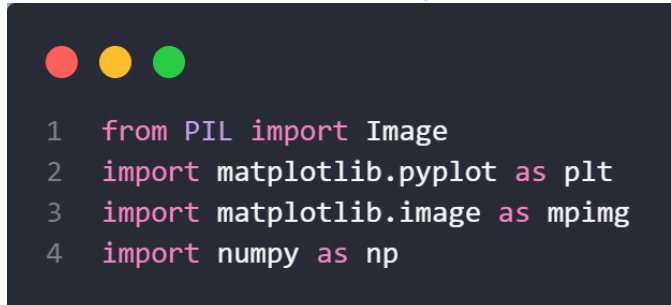
Thuật toán K-means làm việc dựa trên việc tính toán khoảng cách Euclid giữa các điểm dữ liệu và các điểm trung tâm của nhóm để xác định việc gán điểm vào nhóm. Nó cố gắng tối thiểu hóa tổng bình phương khoảng cách giữa các điểm dữ liệu và điểm trung tâm của nhóm để tạo ra các nhóm có tính chất tương tự nhau.

Vấn đề giảm màu có thể được coi là một vấn đề phân cụm. Ví dụ, để chuyển đổi hình ảnh đến 16 màu thì cần tìm vị trí tối ưu của 16 cụm trong không gian RGB đại diện cho không gian đầu vào để lỗi chung sau khi chuyển đổi hình ảnh là giảm thiểu.

- Với cách bước của thuật toán K-means như trên, ý tưởng sử dụng K-means để giảm màu của ảnh với mục tiêu là vẫn bảo toàn được nội dung của ảnh ban đầu:
 1. Chuẩn bị dữ liệu: Đầu tiên, ảnh cần được chuyển đổi thành một ma trận 2D, trong đó mỗi hàng là một vector màu RGB.
 2. Chọn số cluster (k): Số cluster được chọn phụ thuộc vào mức độ giảm màu mong muốn. Số cluster thường sẽ nhỏ hơn số màu gốc của ảnh.
 3. Randomly initialize k centroids: Chọn ngẫu nhiên k điểm trong dữ liệu làm các centroids ban đầu.
 4. Phân loại các điểm: Gán mỗi điểm dữ liệu vào cluster gần nhất dựa trên khoảng cách Euclid giữa điểm và centroids.
 5. Cập nhật centroids: Tính toán lại vị trí của các centroids bằng cách lấy trung bình của tất cả các điểm dữ liệu trong cùng một cluster.
 6. Lặp lại bước 4 và 5 cho đến khi không có sự thay đổi nữa hoặc đạt đến số lần lặp tối đa.
 7. Xác định màu mới: Sau khi đã có các centroids cuối cùng, chúng ta lấy giá trị của mỗi centroid làm màu mới cho các điểm dữ liệu trong cùng một cluster.
 8. Chuyển đổi ảnh: Sử dụng các màu đã xác định ở bước trước để tạo ra ảnh mới đã giảm màu.

3. Mô tả các hàm

Dưới đây là các thư viện và các hàm được sử dụng để hoàn thiện thuật toán bằng python.




```

1  from PIL import Image
2  import matplotlib.pyplot as plt
3  import matplotlib.image as mpimg
4  import numpy as np

```

Hình 1- Các thư viện cần thiết

- Các thư viện sử dụng là: NumPy (tính toán ma trận), PIL (đọc, ghi ảnh), matplotlib (hiển thị ảnh)
 - Sử dụng mảng NumPy cho phép làm việc hiệu quả với ma trận và mảng, đặc biệt là dữ liệu ma trận và mảng lớn với tốc độ xử lý nhanh hơn nhiều lần. Ví dụ như `np.array(...)`: chuyển đổi ảnh đầu vào thành mảng numpy, thay đổi shape cho `np.darray`: `np.reshape(...)`
 - `Image.open(...)`: đọc ảnh
 - `matplotlib.pyplot.imshow(...)`: hiển thị ảnh




```

1 def get_labels(dataSet, centroids):
2     distances=np.linalg.norm((dataSet[:, np.newaxis]-centroids)**2, axis=2)
3     return np.argmin(distances, axis=1)

```

Hình 2 - Hàm get_labels

- Hàm `def get_labels(dataSet, centroids)` thực hiện việc tính toán nhãn của các điểm dữ liệu trong dataset dựa trên các điểm trung tâm centroids của các nhóm.
 - `dataset[:, np.newaxis]` được sử dụng để mở rộng dataset thành một mảng có chiều thứ 3, để có thể tính toán khoảng cách với mỗi điểm trung tâm. `(dataset[:, np.newaxis]-centroids)**2` tính toán bình phương khoảng cách Euclid giữa mỗi cặp điểm dữ liệu và điểm trung tâm.
 - `np.linalg.norm((dataset[:, np.newaxis]-centroids)**2, axis=2)` kết quả là một ma trận chứa khoảng cách giữa mỗi điểm dữ liệu và mỗi điểm trung tâm.
 - `np.argmin(distances, axis=1)`: tìm ra chỉ số của điểm trung tâm có khoảng cách nhỏ nhất tương ứng với mỗi điểm dữ liệu trong `dataSet`. Trả về một mảng chứa chỉ số của điểm trung tâm có khoảng cách nhỏ nhất đối với mỗi hàng của ma trận `distances`.
 - Kết quả của hàm chính là các nhãn của các điểm dữ liệu trong `dataSet`, nhằm chỉ ra điểm trung tâm gần nhất mà mỗi điểm dữ liệu thuộc về.



```

1 def get_centroids(dataSet, Labels, k):
2     centroids = []
3     for j in np.arange(k):
4         idx_j = np.where(np.array(Labels) == j)[0]
5         centroid_j = dataSet[idx_j, :].mean(axis=0)
6         centroids.append(centroid_j)
7     return np.array(centroids)

```

Hình 3 – Hàm get_centroids

- Hàm `def get_centroids(dataSet, labels, k)` được sử dụng để tính toán và trả về tập hợp các centroids dựa trên dữ liệu và nhãn đã cho.
 - Đầu tiên, khởi tạo một biến `centroids` là một danh sách rỗng để lưu trữ các centroid.
 - Sau đó, vòng lặp `for` được thực hiện trong phạm vi từ 0 đến `k` để tạo `k` centroids. Trong mỗi vòng lặp, các điểm dữ liệu có nhãn tương ứng với `j` được lấy thông qua tổ hợp `np.where` và được lưu trữ trong biến `idx_j`. Sau đó, tập dữ liệu chỉ với các điểm có chỉ số trong `idx_j` được truy cập thông qua `dataSet[idx_j,:]` và trung bình của các điểm này được tính toán bằng cách sử dụng hàm `mean(axis=0)`. Kết quả trung bình này được lưu trữ trong biến `centroid_j` và được thêm vào danh sách `centroids`.
 - Cuối cùng, danh sách các centroids được chuyển đổi thành một mảng `numpy` và trả về.

```
1 def k_means(image, k_clusters, max_iter, init_centroids='random'):
```

Hình 4 – Hàm `k_means`


- Hàm `k_means` có các đối số đầu vào là:
 - `image`: hình ảnh đầu vào
 - `k_clusters`: số lượng cụm
 - `max_iter`: số lần lặp tối đa của thuật toán
 - `init_centroids`: phương pháp khởi tạo điểm trung tâm ban đầu, mặc định là `random`

```
1 im_array=np.array(image)
2 m, n = im_array.shape[0]*im_array.shape[1], im_array.shape[2]
3 reshaped_im=im_array.reshape(m, n)
```

Hình 5 – Xử lý ảnh

- `im_array=np.array(image)`: chuyển hình ảnh đầu vào thành một mảng `numpy`.
- Dựa vào mảng `numpy`, tính số hàng `m`, và số cột `n` của mảng hình ảnh.


- Sử dụng reshape để làm phẳng mảng hình ảnh thành một ma trận 2D.



```
1 centroids=np.zeros((k_clusters, n))
```

Hình 6


- Khởi tạo ma trận chứa tọa độ các điểm trung tâm của các cụm.



```
1 for i in range(k_clusters):
2     indexs = np.random.choice(m, 10, replace=False)
3     centroids[i] = np.mean(reshaped_im[indexs], axis=0)
4
```

Hình 7

- Vòng lặp `for i in range(k_clusters)` được sử dụng để khởi tạo các điểm trung tâm ban đầu cho từng cụm. Đây là nơi hàm `np.random.choice` được sử dụng để chọn ngẫu nhiên 10 điểm trong hình ảnh và tính trung bình của chúng để làm điểm trung tâm.



```
1 labels = np.zeros(m)
```

Hình 8

- Khởi tạo mảng `labels` có kích thước `m` hàng (`m` là số điểm dữ liệu trong `reshaped_im`) và giá trị 0 cho tất cả các phần tử để lưu trữ nhãn của mỗi điểm dữ liệu.

```

1 while max_iter > 0:
2     #1. Gán nhãn
3     labels = get_labels(reshaped_im, centroids)
4
5     #2. cập nhật centroids
6     centroids = get_centroids(reshaped_im, labels, k_clusters)
7     max_iter -= 1

```

Hình 9 – Vòng lặp chính của k-means

- Đây là vòng lặp chính của thuật toán K-means. Trong mỗi lần lặp, `get_labels` được sử dụng để xác định nhãn của mỗi điểm dữ liệu dựa trên các điểm trung tâm hiện tại và hàm `get_centroids` được sử dụng để cập nhật vị trí của các điểm trung tâm dựa trên nhãn hiện tại. Ở đây chúng ta sẽ dừng khi số lượng vòng lặp vượt quá `max_iter` hoặc khi các centroids ngừng thay đổi.

```

1 centroids = np.array(centroids)
2 compressed_image=centroids[labels.astype(int), :]

```

Hình 10

- Chuyển ma trận centroids thành một mảng numpy. Sử dụng nhãn đã được xác định để tạo ra một hình ảnh nén, trong đó mỗi điểm ảnh được thay thế bằng giá trị của điểm trung tâm tương ứng.

```

1 compressed_image = compressed_image.astype(np.uint8)
2 compressed_image=compressed_image.reshape(im_array.shape)
3 compressed_image = Image.fromarray(compressed_image)

```

Hình 11

- `compressed_image.astype(np.uint8)`: chuyển ma trận hình ảnh nén thành hình dạng ban đầu của hình ảnh đầu vào.
- `compressed_image=Image.fromarray(compressed_image)`: chuyển đổi mảng hình ảnh nén thành đối tượng hình ảnh.

4. Kết quả

- **Hình 1:** $k_clusters = 3$

Original Image



Compressed - 3 colors



- **Hình 2:** $k_clusters = 5$

Original Image



Compressed - 5 colors



- **Hình 3:** $k_clusters = 7$

Original Image



Compressed - 7 colors



- **Hình 4:** $k_{\text{cluster}} = 10$

Original Image



Compressed - 10 colors



5. Nhận xét kết quả

- **Thời gian chạy của thuật toán:**

Thời gian chạy của thuật toán phụ thuộc vào các yếu tố:

- Kích thước ảnh: Khi ảnh càng lớn, thuật toán sẽ phải xử lý một lượng lớn các điểm ảnh, dẫn đến thời gian chạy càng lâu.
- Số lượng màu cần giảm: Nếu số lượng màu ban đầu trong ảnh lớn, việc giảm số lượng màu này sẽ mất thời gian hơn. Điều này liên quan trực tiếp đến số lượng cụm (clusters) mà thuật toán K-means sử dụng để phân nhóm các điểm ảnh.
- Độ chính xác: Nếu yêu cầu một độ chính xác cao cho kết quả giảm màu, thuật toán sẽ phải thực hiện nhiều vòng lặp để tìm kiếm phân nhóm tối ưu. Điều này cũng ảnh hưởng tới thời gian chạy của thuật toán.
- Khả năng tính toán của máy tính: Thời gian chạy của thuật toán cũng phụ thuộc vào tốc độ xử lý của máy tính. Máy tính có cấu hình cao sẽ thực hiện thuật toán nhanh hơn so với máy tính có cấu hình thấp.

- **Độ chính xác:**

- Số lượng nhóm (clusters): Khi tăng số lượng nhóm, thuật toán sẽ gom cụm các điểm ảnh có màu tương tự hơn và tạo ra một bức ảnh có màu sắc tương đồng hơn. Càng giảm số lượng nhóm, bức ảnh chỉ còn một số màu cơ bản, mất đi các chi tiết nhỏ. Do đó, độ chính xác của K-means phụ thuộc và việc lựa chọn số lượng nhóm phù hợp với ảnh cần xử lý.
- Khởi tạo các điểm trung tâm ban đầu: Thuật toán K-means có thể rơi vào các điểm cực tiểu địa phương, dẫn đến kết quả gom cụm không tối ưu. Để giải quyết vấn đề này, có thể khởi tạo các điểm trung tâm ban đầu dựa trên một thuật toán hoặc phương pháp khác để tìm ra các điểm gần nhất với các màu trong ảnh.
- Đặc điểm của ảnh: Thuật toán K-means thường không hiệu quả khi áp dụng cho các ảnh có những vùng phức tạp hoặc chứa các chi tiết nhỏ. Với những ảnh có nhiều mảnh, hay đối tượng có chi tiết nhỏ, K-means có thể tạo ra hiệu ứng “bể pixel” hoặc mất các chi tiết nhỏ.
- Số lần lặp của thuật toán: số lần lặp lại của thuật toán càng cao thì độ chính xác của ảnh cũng sẽ càng cao. Ví dụ cùng $k_clusters=7$ nhưng khi thay đổi max_iter thì hai bức ảnh dưới đây có độ chính xác khác nhau.



Hình 1 – max_iter=10



Hình 2 – max_iter=1000

6. Tài liệu tham khảo

- [1] Tài liệu thực hành
- [2] https://phamdinhhkhanh.github.io/deepai-book/ch_ml/KMeans.html
- [3] <https://old.cescg.org/CESCG-2007/papers/Brno-Mikolov-Tomas.pdf>
- [4] <https://hackernoon.com/learn-k-means-clustering-by-quantizing-color-images-in-python>
- [5] <https://www.geeksforgeeks.org/image-compression-using-k-means-clustering/>
- [6] <https://blog.vietnamlab.vn/untitled-11/>
- [7] <https://blog.luyencode.net/thuat-toan-phan-cum-k-means/>
- [8] <https://machinelearningcoban.com/2017/01/01/kmeans/>
- [9] <https://www.naftaliharris.com/blog/visualizing-k-means-clustering/>
- [10] https://en.wikipedia.org/wiki/K-means_clustering