TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG - HCM

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN 1

TOÁN ỨNG DỤNG VÀ THỐNG KÊ

< COLOR COMPRESSION >

Lâm Thanh Ngọc - 21127118
Lớp: 21CLC02

Giảng viên:

Vũ Quốc Hoàng

Nguyễn Văn Quang Huy

Lê Thanh Tùng

Phan Thị Phương Uyên

Ngày 18 tháng 7 năm 2023

Mục lục

1 Ý tưởng thực hiện

- Để phân cụm các điểm ảnh trong ảnh theo màu sắc, ta cần khởi tạo K điểm trung tâm (centroids) đại diện cho K màu khác nhau. Có hai cách thường được sử dụng để khởi tạo các điểm trung tâm này một cách ngẫu nhiên:
  + Cách thứ nhất là chọn K màu bất kỳ trong không gian màu của ảnh, ví dụ như RGB. Cách này đơn giản và nhanh chóng, nhưng có thể dẫn đến việc chọn ra những màu không phù hợp với ảnh, hoặc những màu quá gần nhau.
  + Cách thứ hai là chọn K điểm ảnh bất kỳ trong ảnh làm điểm trung tâm ban đầu. Cách này có thể đảm bảo rằng các màu được chọn thuộc về ảnh, và có khả năng phản ánh được sự phân bố của các màu trong ảnh. Tuy nhiên, cách này cũng có thể gặp phải vấn đề khi chọn ra những điểm ảnh nhiễu, hoặc những điểm ảnh quá xa so với các điểm ảnh cùng màu.
- Biểu diễn mỗi điểm màu trong ảnh dưới dạng vector bằng cách sử dụng một ma trận có số hàng bằng số điểm màu và số cột bằng số kênh màu. Với mỗi điểm màu, tính khoảng cách đến K điểm centroids bằng cách sử dụng công thức khoảng cách Euclid và đánh dấu điểm màu thành giá trị của centroids có khoảng cách đến nó ngắn nhất bằng cách sử dụng phép gán nhãn. Bằng cách này, ta có thể phân chia các điểm màu trong ảnh thành K nhóm dựa trên sự tương đồng về màu sắc.
- Sau mỗi vòng lặp, giá trị của centroids sẽ được tính lại và các điểm màu sẽ được cập nhật đến khi số vòng lặp đạt đến số vòng lặp tối đa được truyền vào hoặc các centroids mới trùng với các centroids cũ đã tạo.

2 Mô tả các hàm

2.1 Các hàm thực thi thuật toán K-means

`random_centroids(k, img_1d, init_centroids)`

Chức năng: Chọn k màu cho k centroids từ hình ảnh đã được chuyển đổi thành mảng 1 chiều cho thuật toán phân cụm k-means bằng một trong hai cách chọn màu: chọn màu ngẫu nhiên bất kỳ hoặc chọn màu ngẫu nhiên trong ảnh. Kết quả trả về của hàm là mảng một chiều chứa chỉ số của centroid gần nhất với mỗi điểm ảnh..

Các tham số truyền vào:

- k: số centroids cần chọn.
- img_1d: mảng numpy chứa các điểm ảnh của ảnh một chiều được chuyển đổi từ ảnh gốc bởi hàm convert_1d, có dạng: (số điểm ảnh, số kênh màu).
- init_centroids: chế độ chọn màu gồm: 'random' để chọn màu ngẫu nhiên bất kỳ và 'in_pixels' để chọn màu ngẫu nhiên trong ảnh.

Các hàm hỗ trợ:

- `np.random.randint(low, high=None, size=None, dtype=int)`: Hàm trả về số nguyên ngẫu nhiên trong khoảng *low* đến *high* (bao gồm cả giá trị *low* và *high*). Đối với đoạn chương trình của bài, khi *init_centroids* có giá trị 'random' thì hàm có tác dụng tạo một mảng k phần tử, mỗi phần tử là một số nguyên ngẫu nhiên từ 0 đến 255, tương ứng với giá trị của một kênh màu. Ngược lại, khi giá trị của *init_centroids* là 'in_pixels' thì hàm sẽ chọn ngẫu nhiên từ 0 đến kích thước của ảnh như chỉ số cho mảng 1 chiều biểu diễn ảnh.
- `np.unique(ar, return_index=False, return_inverse=False, return_counts=False, axis=None, *, equal_nan=True)`: Hàm trả về giá trị phần tử độc nhất trong mảng. Trong hàm random_centroids, hàm np.unique được sử dụng nhằm tránh chọn các màu trùng lặp

`init_labels(img_1d, k, init_centroids)`

Chức năng: Hàm random_centroids được sử dụng để khởi tạo các centroids ngẫu nhiên và tính khoảng cách giữa các điểm dữ liệu với các centroids để xác định nhãn để đánh dấu cho từng điểm dữ liệu. Kết quả trả về của hàm là một mảng các màu đã được đánh dấu.

Các tham số truyền vào:

- img_1d: mảng numpy chứa các điểm ảnh của ảnh một chiều được chuyển đổi từ ảnh gốc bởi hàm convert_1d, có dạng: (số điểm ảnh, số kênh màu).
- k: số centroids cần chọn.
- init_centroids: chế độ chọn màu gồm: 'random' để chọn màu ngẫu nhiên bất kỳ và 'in_pixels' để chọn màu ngẫu nhiên trong ảnh.

Các hàm hỗ trợ:

- `numpy.reshape(a, newshape, order='C')`: Hàm trả về một dạng hình ảnh (shape) mới (số nguyên hoặc bộ số nguyên). Với ý tưởng xem một màu là một phần tử và kích thước là dài * rộng, hàm reshape được sử dụng trong đoạn chương trình dùng để chuyển đổi hai mảng img_1d và centroids:
  + Mảng img_1d từ dạng ma trận có số dòng bằng kích thước ảnh, số cột bằng số kênh màu với shape là (kích thước, số kênh màu) thành dạng ma trận có số dòng bằng kích thước ảnh, số cột bằng 1 với shape là (kích thước, 1, số kênh màu).
  + Mảng centroids từ dạng ma trận có số dòng bằng số điểm centroids, số cột bằng số kênh màu với shape là (số điểm centroids, số kênh màu) thành dạng ma trận có số dòng bằng 1, số cột bằng số kênh màu với shape là (1, số điểm centroids, số kênh màu)
- `np.sum(a, axis=None, dtype=None, out=None, keepdims=<no value>, initial=<no value>, where=<no value>)`: Hàm trả về tổng các phần tử theo cột *axis*. Sau khi "đồng dạng" hai mảng img_1d và centroids, ta dùng hàm sum để tính khoảng cách theo công thức khoảng cách Euclid (cộng theo số kênh màu nên axis sẽ lấy giá trị là cột cuối cùng tương ứng với -1)
- `np.argmin(a, axis=None, out=None, *, keepdims=<no value>)`: Hàm trả về vị trí của các giá trị nhỏ nhất theo cột *axis*. Các điểm màu sẽ được đánh dấu bởi màu của centroids với khoảng cách gần nhất tương ứng với giá trị nhỏ nhất trên cột cuối cùng (axis = -1).

`kmeans(img_1d, k_clusters, max_iter, init_centroids)`

Chức năng: Thực hiện thuật toán k-means để phân cụm các điểm ảnh trong một ảnh một chiều, được sử dụng để nén ảnh bằng cách phân cụm số lượng màu sắc xuất hiện trong ảnh. Các điểm ảnh có màu sắc gần nhau có thể được thay thế bởi các điểm ảnh trong cùng một cụm bằng màu sắc của centroid đó. Kết quả trả về là bộ (tuple) chứa hai mảng: centroids (chứa các tâm cụm) và labels (chứa các điểm màu đã được đánh dấu theo centroid gần nhất)

Cách hoạt động:

- Khởi tạo ngẫu nhiên các centroids theo init_centroids bằng cách sử dụng hàm random_centroids.
- Tạo một mảng mới lưu lại các centroids đã khởi tạo.
- Khởi tạo nhãn cho từng điểm ảnh bằng cách gán nhãn của cụm gần nhất theo khoảng cách Euclid bằng cách sử dụng hàm init_labels.
- Cho đến khi đạt được số lần lặp tối đa hoặc các centroids mới bằng các centroids cũ, cập nhật nhãn cho từng điểm ảnh và tính lại centroid như sau:
  + Cập nhật lại nhãn cho từng điểm ảnh bằng cách gán nhãn của cụm gần nhất theo khoảng cách Euclid bằng cách sử dụng hàm init_labels.
  + Tính lại tâm cụm cho từng cụm bằng cách lấy trung bình cộng của các điểm ảnh thuộc cùng cụm.

Các tham số truyền vào:

- img_1d: mảng numpy chứa các điểm ảnh của ảnh một chiều được chuyển đổi từ ảnh gốc bởi hàm convert_1d, có dạng: (số điểm ảnh, số kênh màu).
- k_clusters: số lượng cụm hay số màu mong muốn sau khi nén ảnh.
- max_iter: số lần lặp tối đa khi cập nhật nhãn và tính lại centroid
- init_centroids: chế độ chọn màu gồm: 'random' để chọn màu ngẫu nhiên bất kỳ và 'in_pixels' để chọn màu ngẫu nhiên trong ảnh.

Các hàm hỗ trợ:

- `np.mean(a, axis=None, dtype=None, out=None, keepdims=<no value>, *, where=<no value>)`: Hàm trả về giá trị trung bình cộng theo cột *axis* tương ứng. Giá trị

của centroid thứ i được tính bằng cách lấy trung bình cộng các giá trị trong cùng cụm.

- `numpy.array_equal(a1, a2, equal_nan=False)`: Hàm trả về true nếu hai mảng có các phần tử và hình dạng bằng nhau, ngược lại trả về false. Trong thuật toán kmeans, hàm được sử dụng so sánh hai mảng centroids cũ và mới để tạo lập điều kiện dừng.

2.2 Các hàm chuyển đổi

`convert_1d(img, size)`

Chức năng: Chuyển đổi hình ảnh ban đầu về mảng một chiều lưu trữ kích thước và số kênh màu có dạng (dài * rộng, số kênh màu). Kết quả trả về là mảng một chiều với kiểu dữ liệu 'uint8'

Các tham số truyền vào:

- img: hình ảnh gốc
- size: kích thước ảnh được định nghĩa bằng cách sử dụng hàm init_image

Các hàm hỗ trợ:

- `init_image(img)`: Hàm dùng để khởi tạo kích thước ảnh gốc được truyền vào (tham số img) với dạng: (dài, rộng)

`convert_2d(size, centroids, labels)`

Chức năng: Chuyển đổi mảng centroids và labels thành ảnh với centroids mang giá trị là màu của các tâm cụm và labels là chỉ số của các tâm cụm đó. Kết quả trả về là mảng chứ kích thước ảnh gồm 2 chiều dài, rộng và số kênh màu của ảnh gốc. Đây chính là ảnh đã được phân cụm theo kênh màu.

Các tham số truyền vào:

- size: kích thước ảnh được định nghĩa bằng cách sử dụng hàm init_image
- centroids, labels: các mảng được tính bằng cách sử dụng hàm kmeans.

2.3 Các hàm nhập, xuất

`print_img(imgs, row, col)`

Chức năng: In số lượng hình theo dạng ma trận gồm dòng và cột.

Các tham số truyền vào:

- imgs: mảng các hình cần in
- row: số dòng mong muốn khi xuất hình ảnh
- col: số cột mong muốn khi xuất hình ảnh

`main()`

Chức năng: Yêu cầu người dùng nhập những thông tin cần thiết: tên ảnh, số lượng cụm, số vòng lặp tối đa và phần mở rộng mong muốn khi lưu ảnh. Nếu định dạng ảnh là "RGBA" thì chuyển thành định dạng "RGB".

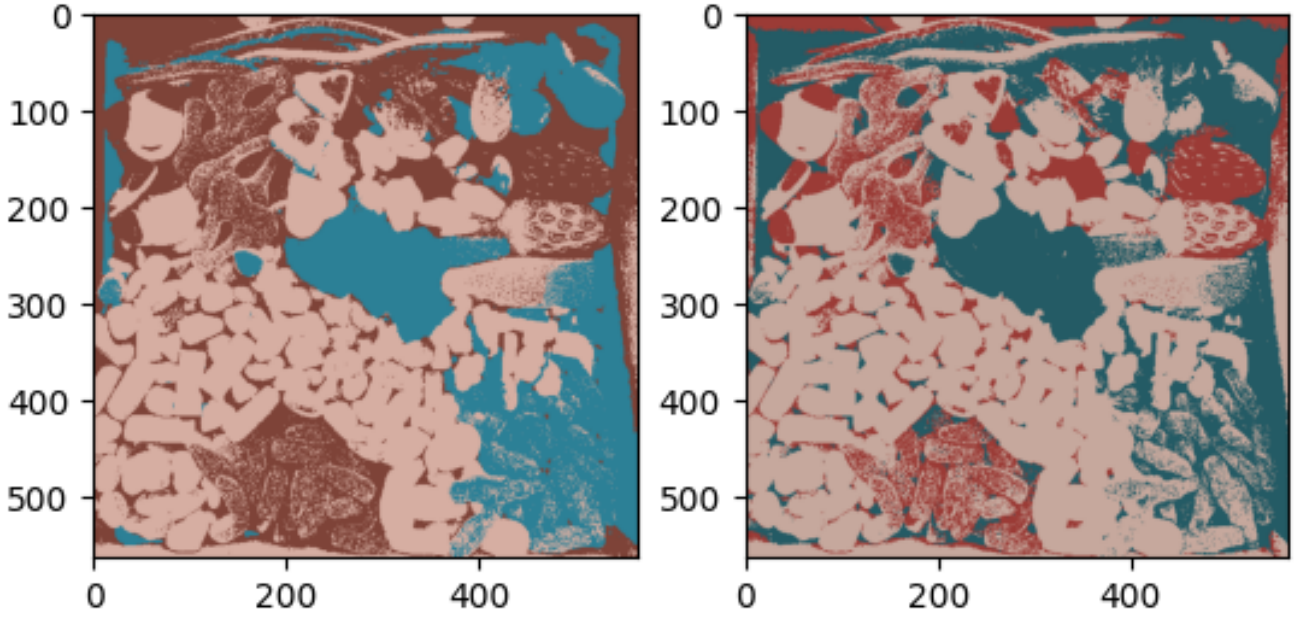
Các hàm hỗ trợ:

- `Image.fromarray(array).save(save_name)`: Hàm dùng để lưu hình ảnh về thiết bị với tên được lưu là "save_name" được đặt mặc định là "new_img" với phần mở rộng do người dùng cung cấp từ một mảng.

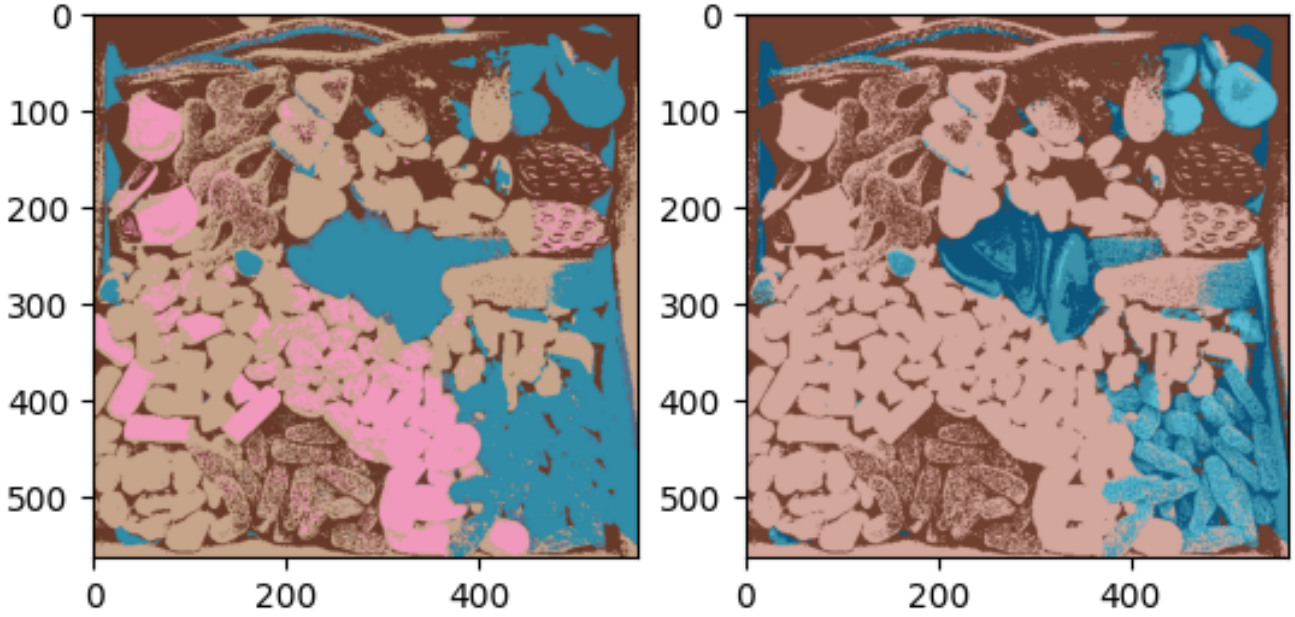
3 Hình ảnh kết quả

Hình 1: Hình ảnh ban đầu

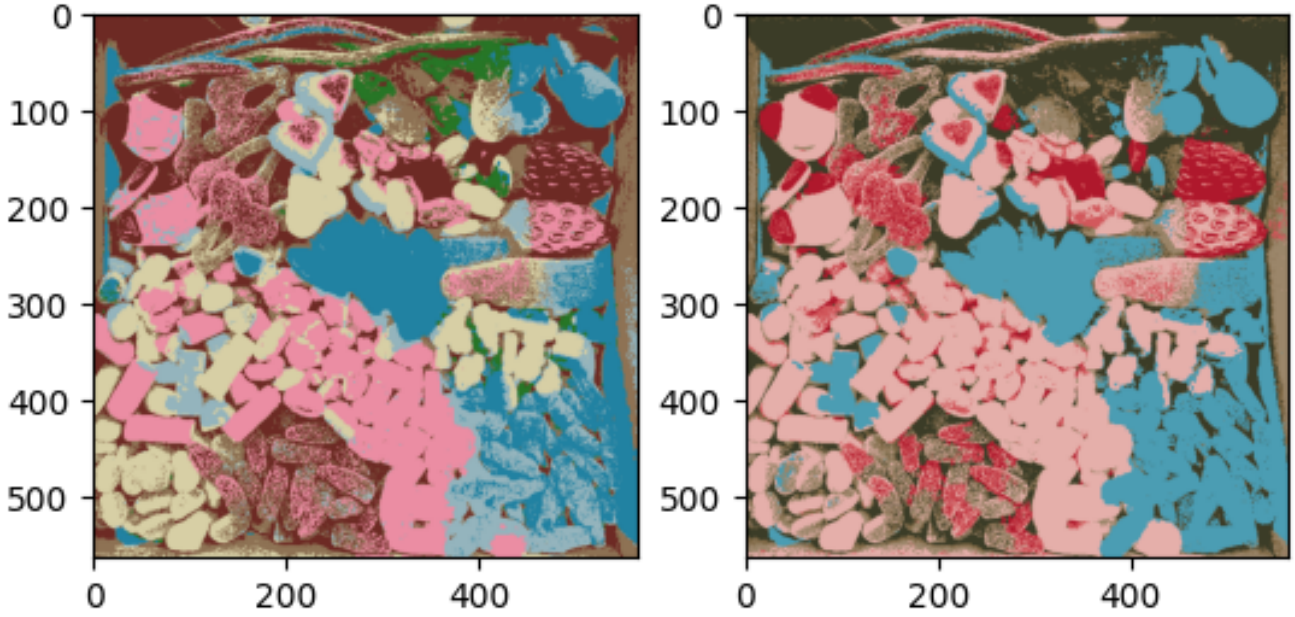
- Thông số ảnh gốc:
  + Kích thước: 81,8KB
  + Chiều rộng: 564 pixels
  + Chiều dài: 564 pixels
- Các hình ảnh kết quả được chạy với số vòng lặp tối đa là 50 và giá trị init_centroids là "random" đối với ảnh bên trái và "in_pixels" đối với ảnh bên phải.



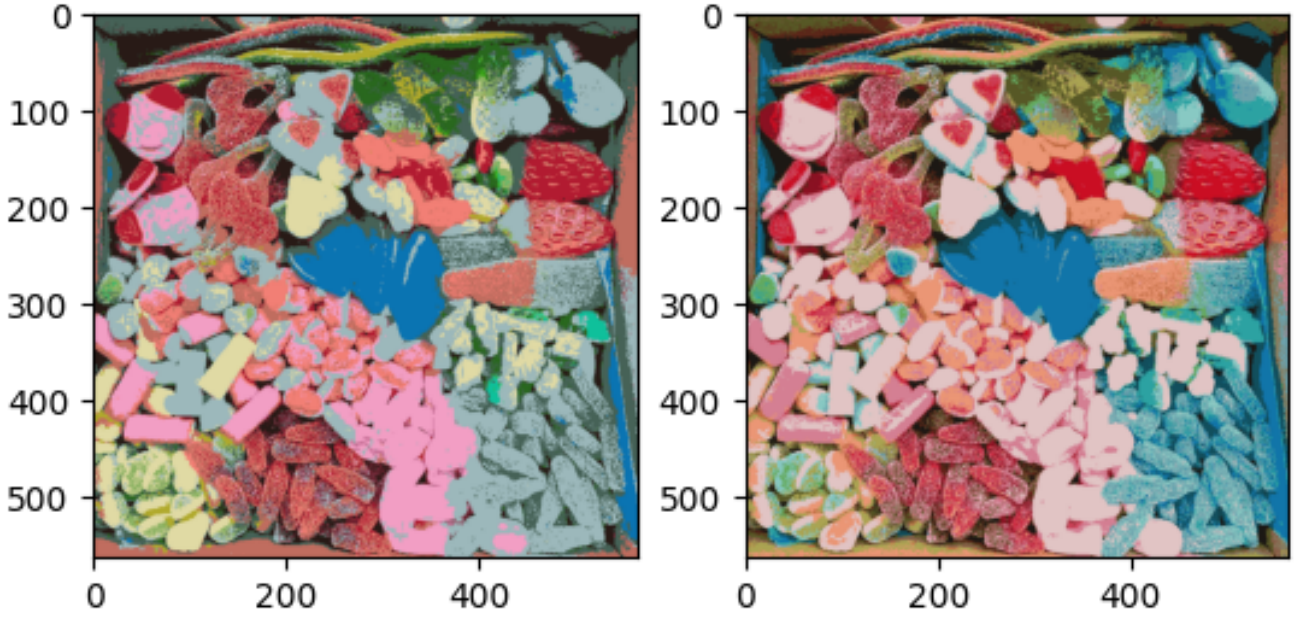
Hình 2: Số cụm được chọn k = 3



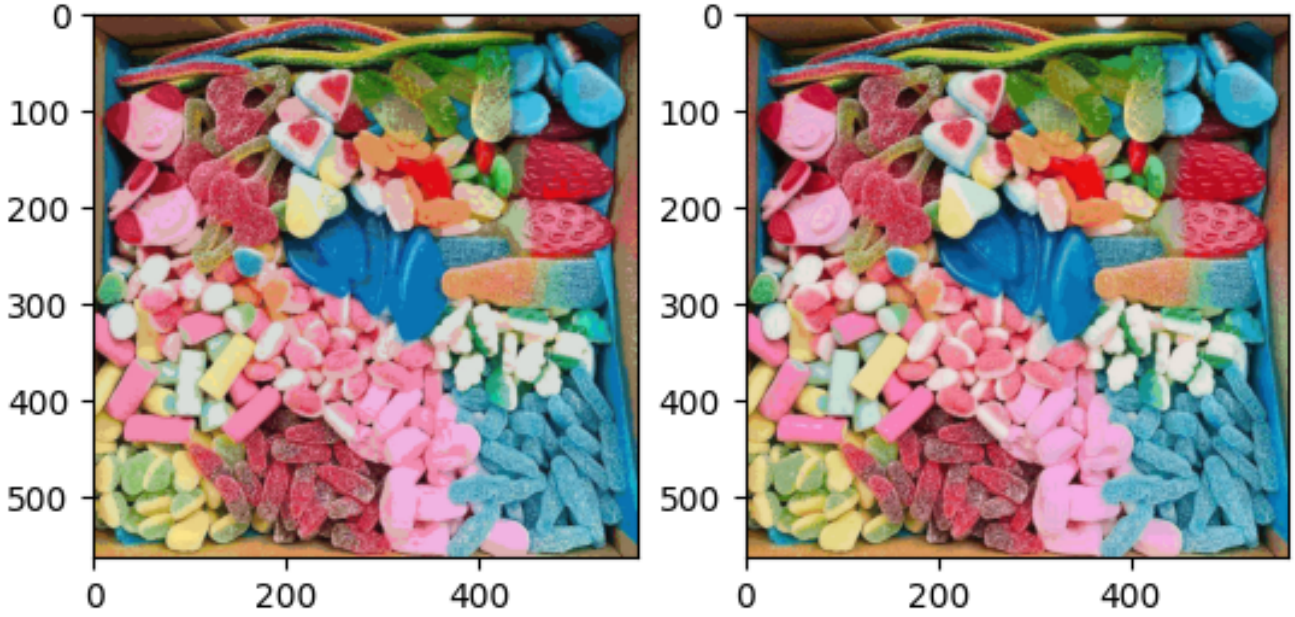
Hình 3: Số cụm được chọn k = 5



Hình 4: Số cụm được chọn k = 7



Hình 5: Số cụm được chọn k = 20



Hình 6: Số cụm được chọn k = 100

4 Nhận xét

- Với số cụm được chọn tăng dần dựa trên giá trị của k, có thể thấy hình ảnh với k càng lớn sẽ càng ít sai lệch về màu. Cụ thể, đối với những k nhỏ như k = {3, 5, 7}, hình ảnh kết quả chỉ giữ lại các sắc màu chiếm đa số trong ảnh gốc và sự phân hóa màu không rõ ràng. Ngược lại, với những k lớn như 20, 100 thì màu sắc đã gần đạt đến gần giống hình ảnh ban đầu.
- Kết quả giữa hai giá trị của tham số init_centroids ("random" và "in_pixels") không có sự chênh lệch quá nhiều về màu sắc

5 Tài liệu tham khảo

- File lab02_project01.ipynb
- Ý tưởng thuật toán K-means
- Các bước thuật toán K-means
- Hàm numpy.random.randint
- Hàm numpy.unique
- numpy.reshape
- numpy.sum
- numpy.argmin
- numpy.mean
- Image.fromarray(array).save(save_name)