

**HO CHI MINH UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY**

-----o0o-----

Applied Mathematics and Statistics for Information Technology

Project 1: Color Compression



Student ID : 23127206
Student name : Lê Nguyễn Nhật Khánh
Class : 23CLC04

Ho Chi Minh City, June 21 2025

Mục lục:

I.	Ý tưởng thực hiện	1
1.	<i>Tổng quan về đồ án.....</i>	1
2.	<i>Cách thức nhập và xuất dữ liệu</i>	1
3.	<i>Mục tiêu và ý tưởng thực hiện.....</i>	1
II.	Chi tiết thực hiện.....	2
1.	<i>Hàm main.....</i>	2
2.	<i>Hàm read_img</i>	2
3.	<i>Hàm convert_img_to_1d.....</i>	2
4.	<i>Hàm kmeans.....</i>	2
5.	<i>Hàm generate_2d_img.....</i>	3
6.	<i>Hàm show_img.....</i>	3
7.	<i>Hàm save_img.....</i>	3
III.	Kết quả và kết luận	4
1.	<i>Kết quả</i>	4
2.	<i>Nhận xét</i>	7
IV.	Tài liệu tham khảo	8
V.	Acknowledgement	8

I. Ý tưởng thực hiện

1. Tổng quan về đồ án

- Trong thời đại số hiện nay, nhu cầu lưu trữ và truyền tải dữ liệu ngày càng tăng. Một trong số đó là dữ liệu dưới dạng hình ảnh; việc sử dụng, lưu trữ hay truyền tải hình ảnh có độ phân giải cao sẽ chiếm nhiều dung lượng bộ nhớ và gây lãng phí bộ nhớ. Vì thế sản sinh ra các bài toán xử lý ảnh nhằm tối ưu hóa việc lưu trữ dữ liệu mà không thay đổi quá nhiều bố cục bức ảnh.
- Một trong những kỹ thuật cơ bản nhất của xử lý ảnh là nén ảnh hay giảm màu ảnh. Kỹ thuật này giảm kích thước lưu trữ của ảnh mà vẫn giữ chất lượng ảnh ở mức chấp nhận được.
- Trong đồ án này, em sử dụng thuật toán phân cụm K-Means để giảm số lượng màu của ảnh xuống một số lượng cố định. Mặc dù đây là một kỹ thuật cơ bản nhưng được áp dụng rộng rãi ví tính hiệu quả của nó.

2. Cách thức nhập dữ liệu và xuất file ảnh theo đường dẫn bất kì

- Cách thức nhập dữ liệu
 - Đầu tiên chương trình sẽ yêu cầu bạn nhập đường dẫn đến ảnh. Bạn có thể chỉ nhập tên file ảnh nếu file ảnh nằm cùng cấp với file .ipynb.
Ex: 'D:/Personal/color_compression/photo/test1.png', 'photo.jpeg', 'abc.png', 'xyz.jpg', '/home/userA/color_compression/test1.png',...
 - Tiếp theo bạn cần nhập số lượng vòng lặp tối đa và số lượng cụm cho chương trình.
 - Và phần nhập cuối cùng trước khi chạy KMeans là bạn cần chọn cách xác định centroids như thế nào. Random hay chọn từ các pixels trong ảnh.
- Cách thức xuất file ảnh theo đường dẫn
 - Đầu tiên bạn cần nhập tên của file ảnh.
Ex: 'abc', 'photo', 'file',...
 - Sau đó chúng ta sẽ chọn đường dẫn nơi file ảnh được lưu.
Ex: 'D:/Personal/color_compression/photo'
 '/home/userA/color_compression'
 - Và cuối cùng là chọn lưu dưới dạng nào: 'png' hay 'pdf'.
 - Chương trình sẽ thông báo đường dẫn cuối cùng mà file ảnh được lưu ra màn hình console.
- *Tất cả nhập xuất đều đã có hướng dẫn trong lúc chạy chương trình.*

3. Mục tiêu và ý tưởng thực hiện

- Mục tiêu của đồ án
 - Tìm hiểu và triển khai thuật toán K-Means trong bài toán xử lý ảnh.
 - Giảm số lượng màu của ảnh nhằm mục tiêu nén ảnh nhưng chất lượng của ảnh vẫn ở mức chấp nhận được.
 - Biết cách sử dụng jupyter notebook
 - Biết sử dụng các thư viện của Python.
- Ý tưởng cốt lõi
 - Đọc ảnh gốc và chuyển thành ma trận các điểm ảnh.
 - Mỗi điểm ảnh là một vector gồm 3 số nguyên biểu diễn cho RGB.
 - Chuyển thành ma trận 1D để dễ xử lý và cho thời gian xử lý nhanh hơn.
 - Sử dụng thuật toán K-Means để gán mỗi pixel về tâm cụm gần nhất và cập nhập giá trị mới của các cụm.
 - Sau khi đã xử lý xong thì tái tạo lại ảnh mới dựa trên các centroids và labels.

II. Chi tiết thực hiện

1. Hàm main

- Mục tiêu của hàm: đây là hàm chính để chạy chương trình, hàm điều phối mọi hoạt động của chương trình.
- Mô tả hàm
 - Hàm sẽ nhận thông tin về ảnh, số vòng lặp, số cụm và cách thức để chọn các centroids.
 - Sau đó sẽ gọi đến hàm 'read_img' để đọc ảnh đầu vào và chuyển ma trận vừa đọc được sang dạng mảng bằng hàm 'convert_img_to_1d'.
 - Gọi K-Means để thực hiện phân cụm màu.
 - Tạo ảnh mới từ các labels và centroids được trả về từ hàm 'kmeans' thông qua hàm 'generate_2d_img'
 - Hiển thị ảnh gốc và ảnh sau nén lên màn hình bằng 'show_img'
 - Sau đó yêu cầu người dùng nhập vào đường dẫn sẽ lưu file, tên file và chọn định dạng file.
 - Cuối cùng là gọi hàm 'save_img' để lưu file theo đường dẫn đã nhập.

2. Hàm read_img

- Mục tiêu của hàm: đọc ảnh từ ổ đĩa, chuyển về định dạng RGB chuẩn.
- Mô tả hàm: Dùng thư viện PIL để mở ảnh và chuyển tất cả ảnh về 3 kênh màu RGB.
- Input của hàm: đường dẫn đến file ảnh cần mở.
- Output của hàm: ảnh ở định dạng PIL.Image với chế độ 3 kênh màu RGB.

3. Hàm convert_img_to_1d

- Mục tiêu của hàm: chuyển ảnh từ dạng ma trận 2D sang dạng 1D để thuận tiện hơn cho thuật toán K-Means.
- Mô tả hàm
 - Dùng '.shape' để lấy ra số cột, hàng và số kênh màu của ảnh gốc.
 - Sau đó dùng '.reshape' để biến thành ma trận 1D với chiều dài bằng số cột nhân với số hàng.
 - Kết quả của hàm trên sẽ là một danh sách các pixel với mỗi pixel là một vector 3 giá trị tương trưng cho RGB.
- Input của hàm: ảnh sau khi được chuyển sang dưới dạng NumPy array.
- Output của hàm: một mảng có kích thức (số pixels, 3).

4. Hàm kmeans

- Mục tiêu của hàm: thực hiện phân cụm các điểm ảnh màu, thuật toán này sẽ nhóm các các pixel có màu gần với nhau thành một cụm màu. Điều này giúp cho bức ảnh sẽ giảm được chi phí lưu trữ mà chất lượng vẫn ở mức chấp nhận được.
- Mô tả hàm
 - Khởi tạo các tâm cụm bằng 'random' hoặc 'in_pixels'.
 - 'random': sinh ngẫu nhiên giá trị của các tâm cụm trong không gian RGB với mỗi giá trị nằm trong khoảng [0, 255].
 - 'in_pixels': chọn ngẫu nhiên từ các pixels của ảnh gốc để làm tâm cụm ban đầu.
 - Gán các điểm ảnh vào cụm gần nhất
 - Chúng ta sẽ tính khoảng cách từ pixel đó đến từng tâm cụm.
 - Gán pixel đó vào cụm có khoảng cách nhỏ nhất.
 - Thay vì sử dụng 2 vòng for để tính khoảng cách thì em đã tối ưu nó bằng cách dùng công thức rút gọn khoảng cách Euclidean.

$$\|x-c\|^2 = \|x\|^2 + \|c\|^2 - 2*x*c$$

- Sử dụng cộng và nhân ma trận để tính đồng thời toàn bộ ma trận khoảng cách giữa pixels và các tâm cụm. Điều này giúp tăng hiệu suất của thuật toán lên đáng kể.
- Cập nhập lại các cụm
 - Sử dụng vòng for để tính trung bình cộng của các pixel thuộc mỗi cụm theo từng kênh màu RGB
 - Cập nhập lại tâm cụm theo giá trị trung bình vừa tính.
- Sau khi đã lặp đủ số vòng hoặc kiểm tra xem nếu như không thay đổi đáng kể thì sẽ dừng chương trình lại và trả về kết quả để tiếp tục xử lý.
- Input của hàm
 - 'img_1d' là danh sách các pixels, mỗi pixel là một vector RGB.
 - 'k_clusters' là số cụm màu cần phân.
 - 'max_iter' là số vòng lặp tối đa.
 - 'init_centroids' là cách khởi tạo tâm cụm dựa trên 'random' hay 'in_pixels'.
- Output của hàm
 - 'centroids' là danh sách các giá trị của các cụm màu sau khi triển khai thuật toán.
 - 'labels' là danh sách chỉ ra các cụm màu mà các pixels thuộc về.

5. Hàm generate_2d_img

- Mục tiêu của hàm: tái tạo lại ảnh dưới dạng ma trận NumPy sau khi đã được xử lý bằng thuật toán kmeans.
- Mô tả hàm
 - Lấy ra kích thước của ảnh gốc.
 - Cập nhập giá trị của các pixels theo từng cụm màu dựa trên 'centroids' và 'labels'.
 - Cuối cùng là reshape lại về dạng ảnh 2D.
- Input của hàm
 - 'shape' là kích thước của ảnh gốc.
 - 'centroids' là danh sách các giá trị của các tâm cụm.
 - 'labels' là danh sách cho biết mỗi pixel ứng với cụm màu nào.
- Output của hàm: ảnh mới dưới dạng NumPy array.

6. Hàm show_img

- Mục tiêu của hàm: hiển thị ảnh bằng thư viện matplotlib.
- Mô tả hàm
 - Dùng 'plt.imshow' để vẽ ảnh.
 - Tắt trục bằng axis để ảnh gọn gàng hơn.
 - Hiển thị ảnh lên màn hình.
- Input của hàm: mảng ảnh 2D dưới dạng NumPy array.
- Output của hàm: hiển thị ảnh lên màn hình.

7. Hàm save_img

- Mục tiêu của hàm: lưu ảnh thành file với định dạng '.png' hay '.pdf' theo đường dẫn được người dùng cung cấp.
- Mô tả hàm
 - Yêu cầu người dùng chọn định dạng sẽ lưu.
 - In đường dẫn cuối cùng lên màn hình.
 - Dùng thư viện Image để chuyển lại thành ảnh rồi lưu vào ổ đĩa.

- Input của hàm
 - ‘img_2d’ là ảnh sau khi xử lý dưới dạng NumPy array.
 - ‘img_path’ là đường dẫn đến vị trí cần lưu.
- Output của hàm: lưu ảnh thành công vào đĩa.

III. Kết quả và kết luận

1. Kết quả

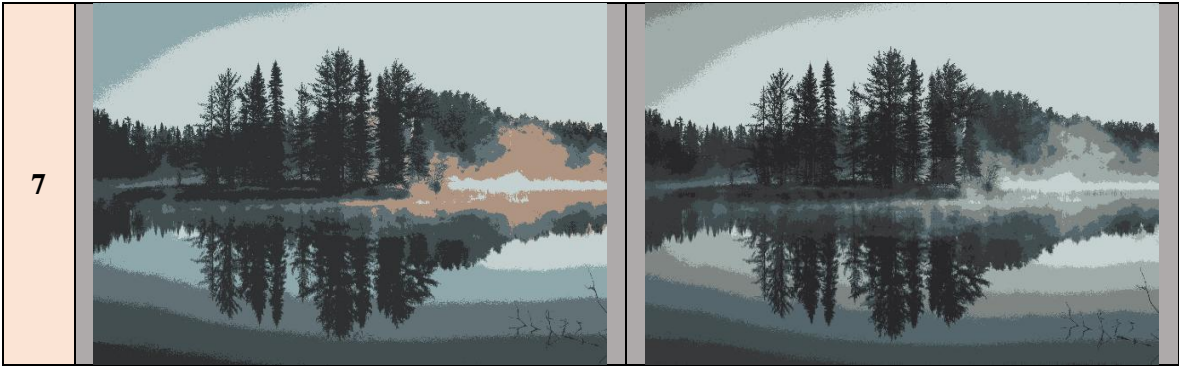
a. Test case 1

- Với ảnh gốc như sau:



- Chúng ta có các ảnh sau khi nén:

K	‘init_centroids’ = ‘random’	‘init_centroids’ = ‘in_pixels’
3		
5		





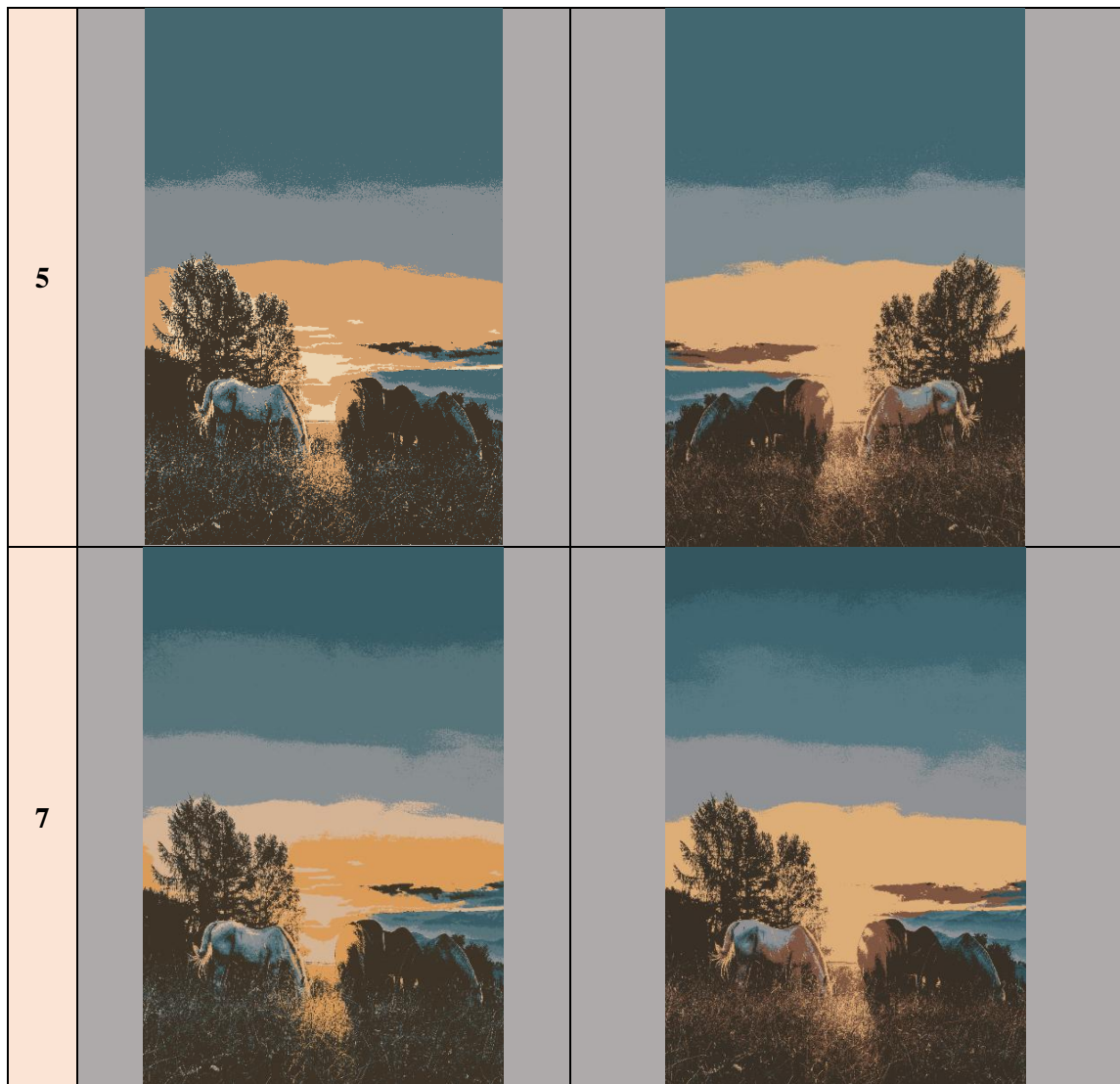
b. Test case 2

- Với ảnh gốc như sau:



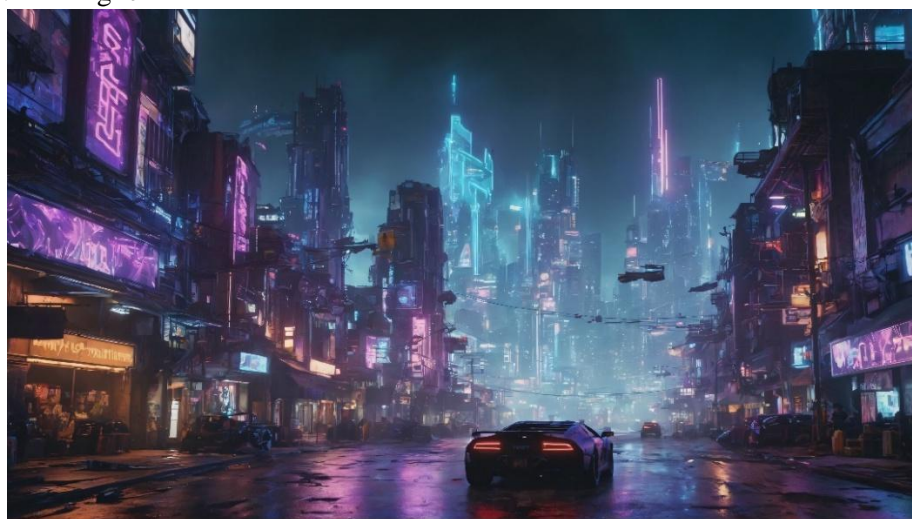
- Chúng ta có các ảnh sau khi nén:

K	'init_centroids' = 'random'	'init_centroids' = 'in_pixels'
3		

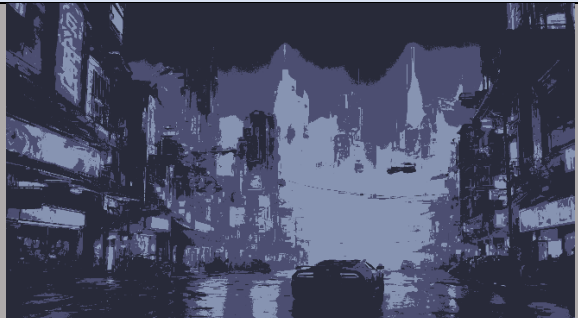
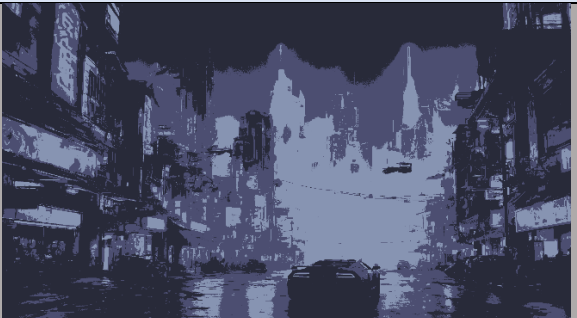






c. Test case 3

- Với ảnh gốc như sau:



- Chúng ta có các ảnh sau khi nén:

K	'init_centroids' = 'random'	'init_centroids' = 'in_pixels'
3		
5		
7		

2. Nhận xét

- Với những ảnh có số lượng màu gốc lớn, thời gian xử lý của thuật toán K-Means sẽ tăng đáng kể do số điểm cần phân cụm nhiều hơn. Mỗi vòng lặp sẽ tốn nhiều thời gian tính khoảng cách và cập nhật centroid.
- Khi số cụm màu càng tăng thì thuật toán cũng trở nên chậm hơn vì cần tính toán nhiều hơn và mất thời gian hơn trong việc cập nhật centroid.
- Với random thì có thể mỗi lần sẽ cho một ảnh khác nhau vì ta không kiểm soát được giá trị được random ra là bao nhiêu.
- Với kĩ thuật xử lý ảnh này, nó sẽ giúp cho ta giảm được số lượng màu, kích thước tổng thể của file ảnh nhưng với chất lượng ảnh vẫn chấp nhận được.

IV. Tài liệu tham khảo

- Geeksforgeeks
 - https://www.w3schools.com/python/numpy/numpy_intro.asp
 - <https://www.geeksforgeeks.org/machine-learning/image-compression-using-k-means-clustering/>
- Github
 - <https://github.com/heroorkrishna/Image-compression-using-K-means-Clustering/blob/master/image%20compression.py>
 - <https://github.com/faithdanghuy/Color-Compression/blob/main/Report.pdf>
 - <https://github.com/kieuconghau/color-compression/blob/master/Document/18127259.pdf>
- Youtube: <https://www.youtube.com/watch?v=shu4pYQb-ps>

V. Acknowledgement

- Đồ án có sự giúp đỡ của bạn Long trong thứ tự nhập các thông tin để xử lý và tiến hành so sánh nhằm kiểm tra độ chính xác cũng như thời gian chạy của thuật toán.
- Đồ án có sự giúp đỡ của bạn Tài trong tối ưu hóa việc tính khoảng cách và gán centroid cho các pixel.
- Bìa của report được phát triển dựa trên report của anh Đăng Huy trong link github thứ 2 ở trên.
- Có sự giúp đỡ của ChatGPT trong việc
 - Tìm hiểu các hàm trong các thư viện cần dùng.
 - Fix bug nhanh chóng hơn.
 - Tạo ảnh để test.
 - Tạo hàm đếm màu, đếm thời gian.
 - Tối ưu hóa code bằng các hàm có sẵn để thuật toán chạy nhanh hơn.