

MSSV : 22127257  
Tên : Phạm Minh Mẫn  
Lớp : 22CLC01

# BÁO CÁO ĐỒ ÁN COLOR COMPRESSION

## Table of content

<b>BÁO CÁO ĐỒ ÁN .....</b>	<b>1</b>
<b>COLOR COMPRESSION .....</b>	<b>1</b>
<b>1/ Giới thiệu chung.....</b>	<b>2</b>
a/ K-means clustering là gì ? .....	2
b/ Nguyên lý hoạt động .....	2
c/ Hình ảnh dưới góc nhìn máy tính .....	2
d/ K-means và giảm màu cho ảnh .....	3
<b>2/ Cài đặt thuật toán K-means .....</b>	<b>3</b>
a/ Các thư viện sử dụng: .....	3
b/ Các hàm phụ trợ : .....	3
c/ Chọn centroids khởi đầu .....	4
d/ Thuật toán k-means .....	4
<b>3/Kết quả chạy chương trình: .....</b>	<b>6</b>
<b>4/ Phụ lục : Max pooling và bài toán thời gian .....</b>	<b>10</b>
a/ Giới thiệu chung .....	10
b/ Cài đặt max pooling .....	11
c/ Nhận xét: .....	14
<b>5/ Kết luận .....</b>	<b>14</b>
<b>Nguồn tham khảo: .....</b>	<b>15</b>

# 1/ Giới thiệu chung

## a/ K-means clustering là gì ?

Phân cụm K-means (K-means clustering) là một phương pháp lượng tử hóa vector dùng để phân các điểm dữ liệu cho trước vào các cụm khác nhau (cluster).

Phân cụm k-means có nhiều ứng dụng, nhưng được sử dụng nhiều nhất trong Trí tuệ nhân tạo và Học máy (cụ thể là Học không có giám sát), chúng cũng được ứng dụng trong phân vùng hình ảnh. [1]

## b/ Nguyên lý hoạt động

Thuật toán k-means sử dụng phương pháp tạo và cập nhật để phân nhóm các dữ liệu cho trước vào các nhóm khác nhau [2]

Mỗi phân vùng dữ liệu gọi là một cluster. Bên trong cluster có một tâm gọi là centroids.

Trong quá trình thực hiện k-means, các centroids được cập nhật liên tục. Điều kiện để dừng thuật toán là khi không có centroids nào thay đổi sau hai lần lặp liên tiếp.

Tuy nhiên, việc đạt kết quả hoàn hảo rất khó và tốn thời gian nên thuật toán sẽ dừng khi đạt đến kết quả gần đúng và chấp nhận được. [3]

## c/ Hình ảnh dưới góc nhìn máy tính

Dưới góc nhìn máy tính, một bức ảnh không chỉ là một hình ảnh thông thường mà còn là tập hợp các dữ liệu số.

Pixel : Là đơn vị cơ bản của hình ảnh. Một ảnh được chia thành nhiều điểm ảnh. Độ phân giải của ảnh (VD : 720 x 1080) chỉ ra số lượng pixel chiều ngang và chiều dọc.

Kênh màu : Mỗi pixel có một giá trị màu. Trong một bức ảnh màu, mỗi pixel thường được biểu diễn bằng ba giá trị tương ứng với ba màu cơ bản: **đỏ (R)**, **xanh lá cây (G)**, và **xanh dương (B)**. Giá trị của mỗi màu thường nằm trong khoảng từ 0 đến 255. (Ví dụ: màu trắng có thể được biểu diễn bằng (255, 255, 255) và màu đen bằng (0, 0, 0)). => Tổng số lượng màu

một ảnh có thể biểu diễn lên tới  $256 \times 256 \times 256 = 16777216$  màu khác nhau.

#### d/ K-means và giảm màu cho ảnh

Với số lượng màu lớn như vậy, việc truyền tải đầy đủ thông tin có thể gặp vấn đề về thời gian và dung lượng. Do đó bài toán sử dụng K-means giảm màu cho ảnh ra đời.

Bài toán này dựa trên ý tưởng sử dụng K-means, phân nhóm các màu lại, giảm tải số lượng màu trong một ảnh nhưng vẫn giữ được tương đối nội dung ban đầu.

## 2/ Cài đặt thuật toán K-means

Sau đây là báo cáo về phần cài đặt thuật toán K-means clustering.

#### a/ Các thư viện sử dụng:

+ Numpy : Đây là thư viện cốt lõi cho scientific computing, nó chứa một đối tượng mảng n chiều mạnh mẽ, hữu ích trong đại số tuyến tính, random number capability, ... . [4]

+ PIL : Thư viện dùng để xử lý hình ảnh.

+ Matplotlib : Một thư viện vẽ đồ thị bằng ngôn ngữ lập trình python.

#### b/ Các hàm phụ trợ :

+ def read\_img(img\_path) : Đọc hình ảnh từ đường dẫn, giá trị trả ra là ma trận các pixel của ảnh. Mỗi pixel bao gồm giá trị 3 màu RGB.

+ def show\_img(img\_2d, title = "") : In hình ảnh lên console dưới dạng đồ thị, ngoài ra còn có trục height, width để ta xem số lượng các pixel có trong ảnh

+ def save\_img(img\_2d, save\_name, file\_format = 'jpg') : Lưu hình ảnh từ ma trận pixel, có chọn save\_name và định dạng (hỗ trợ các định dạng jpg, png, jpeg, pdf.)

+ def convert\_img\_to\_1d(img\_2d) : Sử dụng numpy chuyển từ ma trận pixel thành một mảng các vector RGB.

+ def mean(A) : Sử dụng numpy tính trung bình của một mảng.

### c/ Chọn centroids khởi đầu

Thuật toán k-means khởi đầu bằng việc chọn k centroids ngẫu nhiên.

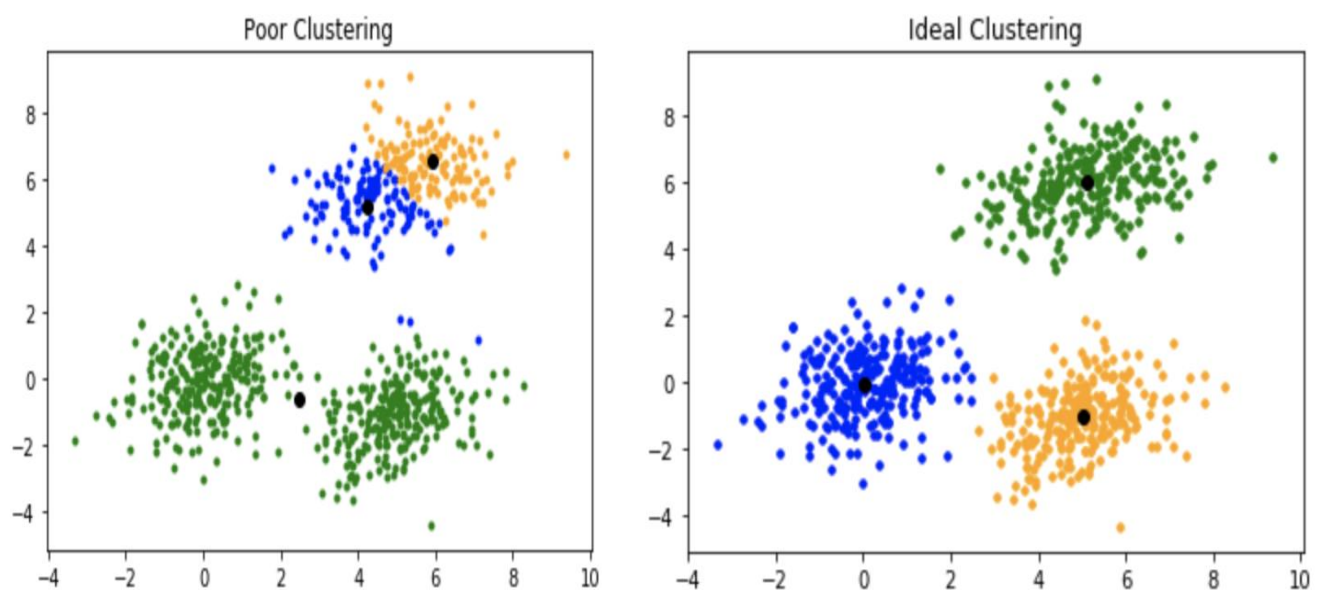
Có vài cách để thực hiện khởi tạo :

+ def set\_centroids\_random(data, k, min = 0, max = 255) : Generate ngẫu nhiên giá trị của 3 màu RGB (từ 0 tới 255) cho k centroids.

+ def set\_centroids\_initpix(data, k) : Chọn k centroids trực tiếp từ bên trong hình ảnh, một cách ngẫu nhiên.

+ def set\_centroids\_plus(data, k) : Đây là cách chọn centroids trong phiên bản mở rộng của k-means (k-means ++). Thuật toán này chọn centroids đầu tiên một cách ngẫu nhiên, các centroids còn lại được chọn sao cho xa nhất với các centroids đã có.

Việc chọn các centroids khởi đầu có ảnh hưởng không nhỏ tới kết quả đầu ra của k-means.



Chọn centroids không tốt gây ra các cluster không tốt [5]

### d/ Thuật toán k-means

Trong bài làm, quá trình thực hiện kmeans nằm trong ba hàm sau :

+ def run\_iterations(data,k, iterations = 100, init\_centroids = 'random') :

Đây là cốt lõi của thuật toán k-means, gồm các bước:

Bước 1 : chọn centroids thông qua một trong 3 cách (random, in\_pixels, plus)

Bước 2 : Tính khoảng cách để dán nhãn cho cho tất cả điểm dữ liệu dựa trên bộ centroids (một điểm sẽ nhận centroids gần nó nhất làm tâm).

Bước 3 : Tính mean theo từng cluster. Nếu mean khác centroids ban đầu, ta gán mean thành centroid mới

Bước 4 : Lặp đi lặp lại cho tới khi mean tính ra được xấp xỉ centroids (tức giá trị chấp nhận được).

+ def iterative\_kmeans(data, k, iterations=100, init\_centroids = "random", attemps = 1) : Hàm lặp đi lặp lại k-means để chọn ra centroids tốt nhất.

Tiêu chí đánh giá một bộ centroids tốt :

Withing cluster sum of square : Khái niệm đo lường mức độ tương tự hay khác biệt giữa các điểm trong một cụm dữ liệu (cluster).

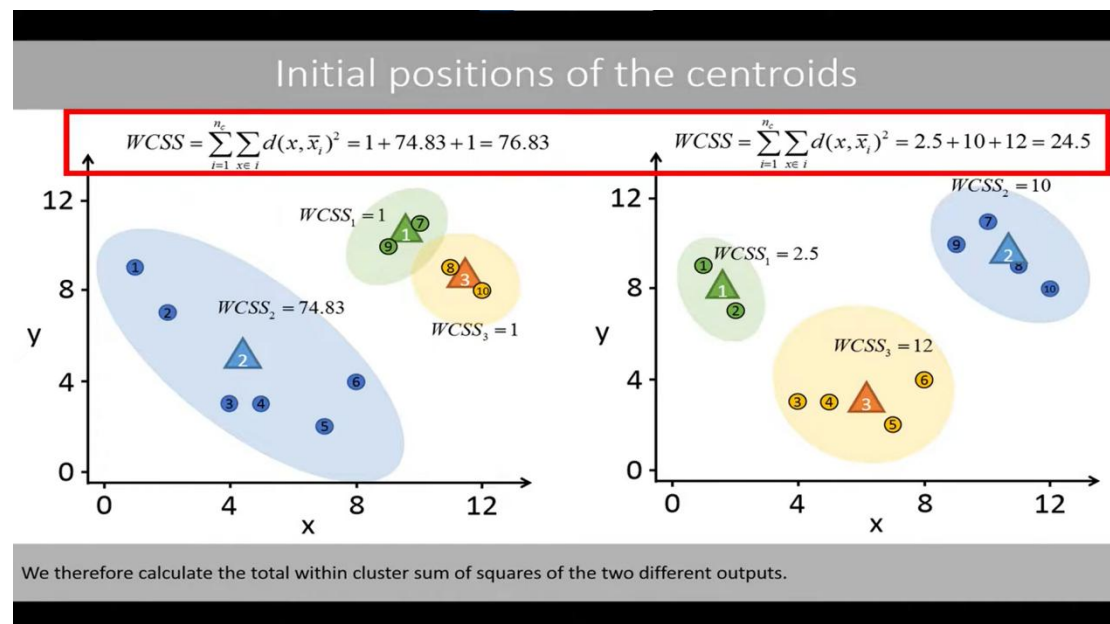
$$WCSS = \sum_c \sum_x |x_i - \mu_i|$$

Với :

c : Một cluster

x : Một điểm dữ liệu trong cụm.

$\mu$  : Khoảng cách từ điểm đó tới centroid của cụm.



WCSS càng nhỏ thì dữ liệu trong cụm càng gần nhau, tức cụm đó càng chặt chẽ. Hình bên phải có WCSS nhỏ hơn bên trái, các điểm trong cụm gần centroid hơn và các centroid xa nhau hơn[6]

+ def kmeans(img\_1d, k\_clusters, max\_iter, init\_centroids='random') :  
Đây là hàm bao gồm toàn bộ quá trình chạy kmeans, trả về bộ centroids  
và nhãn labels.

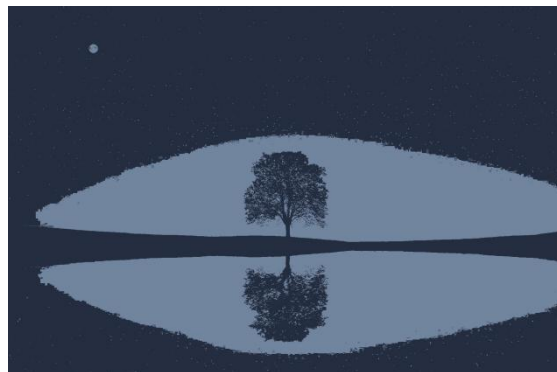
### 3/Kết quả chạy chương trình:



Ảnh 1

3 cluster

random



In\_pixel



plus



5 cluster


random



In pixels





plus	
------	--

7 cluster

random



In pixels



plus

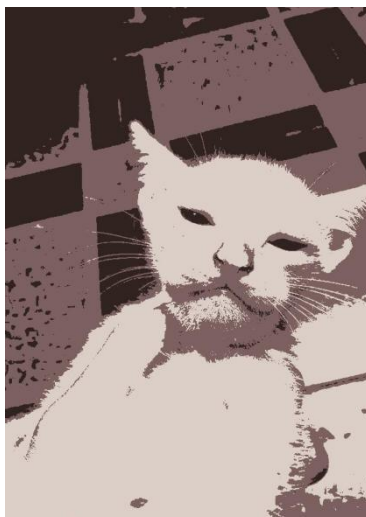






Ảnh 2

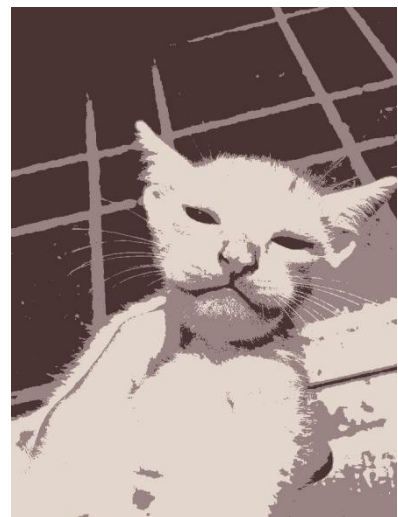
3 clusters



random



In pixels



plus

### 5 clusters



random



in pixels



plus

### 7 clusters



random



in pixels



plus

Nhận xét : Nhìn chung kết quả thu được tương đối tốt (khi so sánh với sk learn). Trong đó, các centroids được khởi tạo in pixels và plus cho kết quả tốt hơn random.

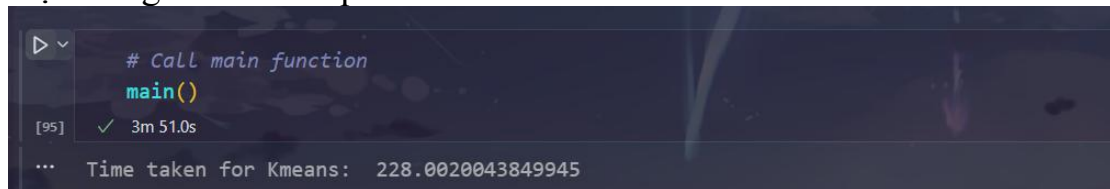
## 4/ Phụ lục : Max pooling và bài toán thời gian

### a/ Giới thiệu chung

Trong machine learning, thời gian là một yếu tố quan trọng ảnh hưởng đến quy trình xây dựng và triển khai mô hình, trong đó quan trọng nhất phải kể đến là thời gian huấn luyện (Training time).

Trong lĩnh vực học máy liên quan đến xử lý hình ảnh và thị giác máy tính, các mô hình phải đối mặt với một lượng lớn hình ảnh, mỗi hình ảnh có thể lên tới hơn 10.000 pixel và mỗi pixel có tới  $256^3$  kênh màu khác nhau.

Mô hình k-means trong color compression có thể xử lý 3, 5, 7 clusters trong vài giây. Tuy nhiên, khi số lượng cluster nhiều lên (30, 50, 70 clusters), cần nhiều iterations với độ chính xác cao hơn ( $10^{-4}$ ,  $10^{-5}$ ), kích thước hình ảnh lớn hơn, hay đơn giản cần xử lý nhiều hình ảnh, cách biệt thời gian trở nên quá lớn.

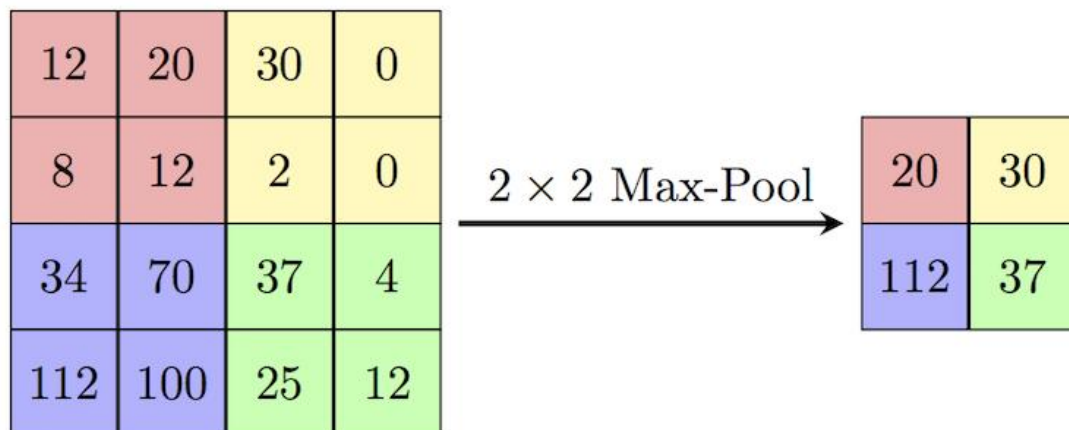


```
# Call main function
main()
[95] ✓ 3m 51.0s
... Time taken for Kmeans: 228.0020043849945
```

Thời gian giảm màu 50 clusters lên tới hơn 3 phút, không thích hợp khi cần xử lý nhiều ảnh

Do đó, việc tối ưu hóa thời gian là cần thiết.

Max pooling là một kỹ thuật phổ biến giúp giảm kích thước của các đặc trưng (ở đây là pixel trong hình ảnh). Điều này giảm số lượng tham số tính toán bên trong mô hình



Minh họa max pooling [7]

#### b/ Cài đặt max pooling

def get\_sub\_matrix(A, irow, icol, row\_size, col\_size) : Hàm lấy ma trận con từ vị trí irow, icol. Ma trận con ở đây là pool window.

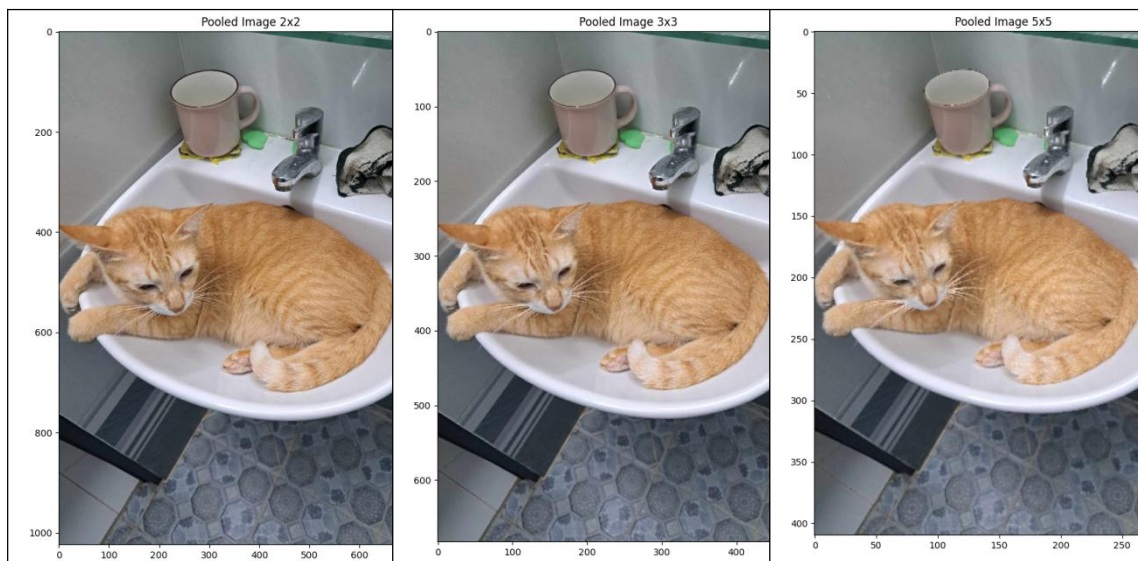
def max\_color(window) : Trong một pool window, lấy ra giá trị max của 3 màu red, green, blue.



`def max_pooling(A, pool_size = 2, stride = 2)` : Hàm thực hiện max pooling, kích thước mặc định là 2 và bước trượt là 2.



Ảnh gốc, khoảng 1400 x 2000 pixels



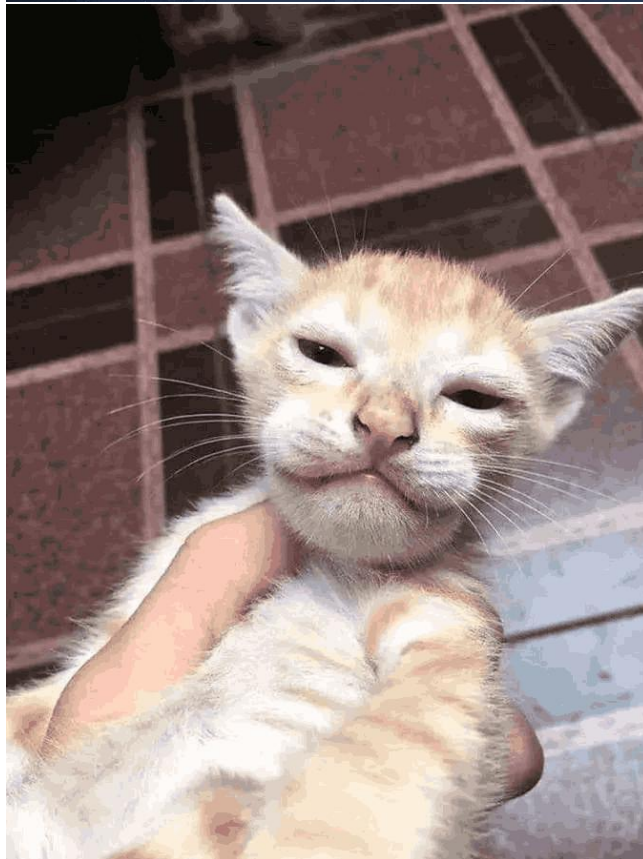
Ảnh sau khi pool 2x2, 3x3, 5x5, ta nhận ra số lượng pixel đã giảm.

Vì số lượng pixel giảm đáng kể, thời gian thực hiện k-means cũng giảm theo.

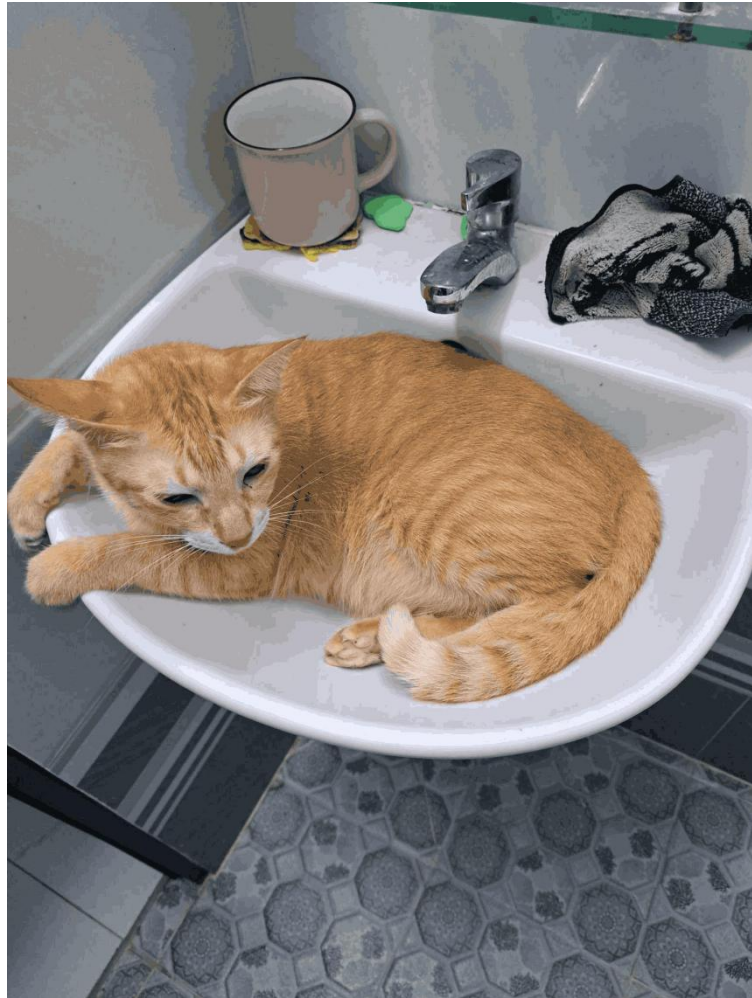
50 clusters, thời  
gian chạy k-means :  
12 giây



50 clusters, thời  
gian chạy k-means :  
10 giây



50 cluster, thời gian  
chạy k-means:  
33 giây



#### c/ Nhận xét:

Ưu điểm :

- +Thời gian train giảm đáng kể.
- +Kết quả đưa ra tương đối tốt (khi so sánh với sk learn).

Nhược điểm :

- +Ảnh bị max pooling có thể bị mất thông tin, không phù hợp cho hình ảnh phức tạp.
- +Có thể gây ra hiện tượng dải màu (như hình 1) làm hình mất tự nhiên.

Nhận xét : Nhìn chung max pooling kết hợp với k-means cho ra kết quả tương đối tốt, hơn nữa còn giải quyết được vấn đề thời gian chạy (từ vài phút xuống vài giây).

## 5/ Phần kết

Đồ án color compression sử dụng k-means là một đồ án hay, giúp em học được nhiều điều liên quan đến machine learning.

Trân trọng cảm ơn các thầy thực hành đã tận tình hướng dẫn và chỉ bảo trong suốt thời gian học môn Toán ứng dụng này.  
Cảm các thầy đã dành thời gian đọc bài báo cáo này, xin chúc thầy nhiều sức khỏe.

Nguồn tham khảo:

[1] : Phân cụm k-means, giới thiệu, ứng dụng

[https://vi.wikipedia.org/wiki/Ph%C3%A2n\\_c%E1%BB%A5m\\_k-means](https://vi.wikipedia.org/wiki/Ph%C3%A2n_c%E1%BB%A5m_k-means)

[2] : Phân cụm k-means, mô tả chung

[https://vi.wikipedia.org/wiki/Ph%C3%A2n\\_c%E1%BB%A5m\\_k-means](https://vi.wikipedia.org/wiki/Ph%C3%A2n_c%E1%BB%A5m_k-means)

[3] : Phân cụm k-means, mô tả chung

[https://vi.wikipedia.org/wiki/Ph%C3%A2n\\_c%E1%BB%A5m\\_k-means](https://vi.wikipedia.org/wiki/Ph%C3%A2n_c%E1%BB%A5m_k-means)

[4] : Python Numpy tutorial, Python numpy là gì

<https://viblo.asia/p/python-numpy-tutorial-hoc-numpy-arrays-voi-cac-vi-du-ORNZqjPnl0n>

[5] : ML| Kmean++ algorithm, poor clustering and ideal clustering

<https://www.geeksforgeeks.org/ml-k-means-algorithm/>

[6] : k-means clustering explained, 7:30

[https://www.youtube.com/watch?v=4E\\_DFMt60rc](https://www.youtube.com/watch?v=4E_DFMt60rc)

[7] : Papers with code, max pooling

<https://paperswithcode.com/method/max-pooling>

Nguồn ảnh : sưu tầm.

THE END  
THANK YOU FOR READING