**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN 1 - TOÁN ỨNG DỤNG**

**| Đề Tài |**

**COLOR COMPRESSION**

**| Giáo viên hướng dẫn |**

**ThS. Phan Thị Phương Uyên**

**Thành Phố Hồ Chí Minh – 2022**

# **MỤC LỤC VÀ GIỚI THIỆU**

[**MỤC LỤC VÀ GIỚI THIỆU** 2](#_Toc106564609)

[**Giới thiệu:** 2](#_Toc106564610)

[**Ý TƯỞNG VÀ MÔ TẢ** 3](#_Toc106564611)

[**Ý tưởng chính:** 3](#_Toc106564612)

[**Mô tả các hàm:** 3](#_Toc106564613)

[**HÌNH ẢNH KẾT QUẢ VÀ NHẬN XÉT** 5](#_Toc106564614)

[**Nhận xét**: 6](#_Toc106564615)

[**THAM KHẢO** 7](#_Toc106564616)

## **Giới thiệu:**

Đồ án được thực hiện bởi sinh viên:

* Tên: Phạm Nguyễn Cao Cường
* MSSV: 20127457
* Đại học Khoa học tự nhiên - ĐHQG TP Hồ Chí Minh.
* Khoa Công nghệ thông tin

# **Ý TƯỞNG VÀ MÔ TẢ**

## **Ý tưởng chính:**

Thuật toán K-means hay còn được gọi là thuật toán phân cụm k-means (k-means clustering) là một thuật toán giúp ta phân vùng, phân loại dữ liệu dựa trên một danh sách các tiêu chí là các “tâm” (centroid).

Lấy ví dụ trên mặt phẳng *Oxy*, ta có *n* điểm ngẫu nhiên nằm rải rác trên mặt phẳng. Ta muốn phân loại các điểm đó về một số tập hợp với mỗi tập là các điểm nằm gần nhau (cụ thể là *3 tập hợp*).

* Chọn *k* “tâm” (centroid) bất kỳ hoặc để làm các cột mốc đánh giá dữ liệu, hay nói cách khác là xác định khoảng cách của các điểm để xem điểm đó thuộc tập hợp nào (gần với tâm của tập hợp).
* Tính toán để xác định khoảng cách của các điểm tới tâm gần nhất.
* Với *x* điểm của mỗi tập hợp, ta tìm trung vị (hoặc trung bình cộng) để tìm ra centroid mới.
* Thực hiện lại bước tính toán để phân loại lại *n* điểm về các tập hợp với centroid mới cho tới khi centroid mới và centroid cũ không quá khác biệt.

Trong đồ án thì với centroid chính là một điểm ảnh bất kỳ, ta tính toán để đưa các điểm ảnh của ảnh cần nén (color compression) về thành màu của các centroid, khi đó số màu cần lưu thấp hơn nhưng vẫn giữ được nội dung đại khái của ảnh.

Lưu ý: Sử dụng phép tính chuẩn giữa 1 điểm ảnh và centroid để xác định điểm ảnh và centroid có “gần” nhau không.

## **Mô tả các hàm:**

* Hàm get\_data(): Lấy dữ liệu ảnh bằng thư viện PIL, trả về mảng numpy chứa dữ liệu các điểm ảnh của ảnh.
* Hàm get\_centroids(): Tạo *k\_cluster* các centroid bằng 2 cách là tạo giá trị ngẫu nhiên hoặc chọn một điểm ảnh bất kỳ trong ảnh thông qua biến **init**.
* Hàm assign\_label(): Lưu *index* của các centroid cho từng điểm ảnh là *index* của centroid gần điểm ảnh đó nhất trong tất cả các centroid thông qua một mảng.
* Hàm update\_centroid(): Cập nhật lại các centroid[*k*] bằng cách tìm trung vị của tập hợp các điểm ảnh gần centroid[*k*] thông qua hàm np.median().
* Hàm converge\_check(): Kiểm tra sự khác biệt giữa các centroid cũ và mới có vượt quá dung sai tuyệt đối (tolerance) E hay không, nếu không có nghĩa là không có ý nghĩa gì khi tiếp tục cập nhật các centroid. Đây chính là điều kiện dừng của hàm kmeans().
* Hàm kmean(): Sử dụng các hàm hỗ trợ để thực thi thuật toán k-means, trả về danh sách *k\_cluster* centroid và mảng lưu trữ index của centroid mà các điểm ảnh thuộc về (mảng labels).
* Hàm save\_result(): Tạo dữ liệu ảnh sau khi nén thông qua mảng centroid và mảng labels sau đó sử dụng hàm figimage() của thư viện PIL để lưu ảnh vào dữ liệu của lớp plt sau đó sài hàm savefig() để xuất file ảnh với các định dạng do người dùng chọn.

# **HÌNH ẢNH KẾT QUẢ VÀ NHẬN XÉT**

A picture containing text, outdoor, several

Description automatically generatedA picture containing text

Description automatically generated

Hình 2. Ảnh kết quả với k = 3

Hình 1. Ảnh màu gốc

A picture containing text, building

Description automatically generatedA picture containing building

Description automatically generated

Hình 4. Ảnh kết quả với k = 7

Hình 3. Ảnh kết quả với k = 5

- Ảnh kết quả cho ra điều sử dụng cách chọn centroid là ‘in\_pixels’ để dễ so sánh với ảnh gốc.

## **Nhận xét**:

Ảnh kết quả nhận được vẫn đảm bảo được nội dung tổng thể của ảnh gốc với số lượng màu ít hơn, khi k\_cluster càng tăng thì ảnh sẽ càng sinh động, nhiều chi tiết và gần với ảnh gốc hơn.

So sánh với kết quả khi sử dụng hàm K-Means trong scikit-learn, ảnh thu được khá tương đồng khi số lượng *k\_cluster* nhỏ, nhưng có một số lần chạy kết quả cho ra khác so hàm của scikit-learn (hoặc khi số lượng *k\_cluster* centroid lớn kết quả cho ra có khác biệt rõ so với hàm của scikit-learn) bởi vì lúc khởi tạo *k\_cluster* centroid đầu ta sử dụng hàm random nên có thể chọn trúng các centroid khá gần nhau hoặc khởi tạo ngẫu nhiên các centroid không quá khác biệt, nên sau đó, việc gom nhóm cũng như cập nhật centroid bị sai số và kết quả cuối cùng khó giữ được đúng màu ban đầu.

Chỉ cần cập nhật thuật toán chọn các centroid có chủ đích thì kết quả cho ra sẽ tốt hơn.

# **THAM KHẢO**

*A demo of K-Means clustering on the handwritten digits data*. (n.d.). Retrieved from scikit-learn.org: https://scikit-learn.org/stable/auto\_examples/cluster/plot\_kmeans\_digits.html#sphx-glr-auto-examples-cluster-plot-kmeans-digits-py

anuragnaren2108. (2020, May 26). *Matplotlib.pyplot.savefig() in Python*. Retrieved from geeksforgeeks.org: https://www.geeksforgeeks.org/matplotlib-pyplot-savefig-in-python/

*Bài 4: K-means Clustering*. (2017, January 1). Retrieved from machinelearningcoban.com: https://machinelearningcoban.com/2017/01/01/kmeans/

Bharadwaj, A. (2019, May 13). *Image compression using K-means clustering*. Retrieved from geeksforgeeks.org: https://www.geeksforgeeks.org/image-compression-using-k-means-clustering/

Đức, K. V. (2021, April 12). *PROJECT 1 : Color Compression*. Retrieved from github.com: https://github.com/kvmduc/applied-math/blob/main/Applied%20Math/Lab02/TO%C3%81N%20%E1%BB%A8NG%20D%E1%BB%A4NG%20V%C3%80%20TH%E1%BB%90NG%20K%C3%8A.md

*sklearn.cluster.KMeans*. (n.d.). Retrieved from scikit-learn.org: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html