TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG – HCM
KHOA CÔNG NGHỆ THÔNG TIN

----- ☙ 🕮 ❧ -----

BÁO CÁO ĐỒ ÁN
PROJECT 1: COLOR COMPRESSION

Môn học: Toán ứng dụng và thống kê cho Công nghệ thông tin

Giảng viên:	Vũ Quốc Hoàng
	Nguyễn Văn Quang Huy
	Nguyễn Ngọc Toàn
	Phan Thị Phương Uyên
Sinh viên thực hiện:	Dương Trung Nghĩa
MSSV:	22127293
Lớp:	22CLC05

Thành phố Hồ Chí Minh, năm 2024

MỤC LỤC

NHẬN XÉT CỦA GIẢNG VIÊN

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Giảng viên

Ý TƯỞNG THỰC HIỆN

1. Giới thiệu đề tài

- Ảnh màu "RGB" là loại ảnh phổ biến, trong đó mỗi điểm ảnh chứa thông tin về ba kênh màu: R (đỏ), G (xanh lá), B (xanh dương). Do đó, một ảnh RGB có thể hiển thị lên đến 256^3 = 16,7 triệu màu. Với số lượng màu sắc khổng lồ này, việc lưu trữ ảnh có thể tốn nhiều dung lượng. Vì vậy, vấn đề mà thầy/cô đặt ra là làm thế nào để giảm số lượng màu sắc trong ảnh mà vẫn giữ được nội dung ảnh càng nguyên vẹn càng tốt.

2. Thuật toán lựa chọn

- Để giảm số lượng màu sắc, cần tìm ra các màu đại diện có thể thay thế cho một nhóm màu tương tự. Dưới sự hướng dẫn của thầy/cô, thuật toán "K-Means" sử dụng trung bình của các điểm màu để phân chia n điểm ảnh vào k cụm màu sao cho các điểm trong cùng một cụm có đặc tính màu sắc khác biệt ít nhất (tương đồng với nhau nhất). Ngoài ra, còn có thuật toán "K-Medians" sử dụng trung vị để xác định "centroids". Ở đây, đồ án được thực hiện với thuật toán "K-Means" vì dễ hiểu và dễ triển khai.

3. Thư viện sử dụng

- NumPy (tính toán ma trận).
- PIL (đọc, ghi ảnh).
- matplotlib (hiển thị ảnh).

4. Các bước thực hiện

- B1) Đọc ảnh từ img_path truyền vào, trả về mảng 3D numpy (height, width, channels).
- B2) Chuyển đổi ảnh từ kích thước mảng 3D (height, width, channels) sang 2D (height * width, channels).
- B3) Khởi tạo centroids ngẫu nhiên với số lượng k_clusters truyền vào (với "random" chọn ngẫu nhiên giá trị từ [0,255], với "in-pixel" chọn ngẫu nhiên giá trị trong các pixel màu).
- B4) Tính toán khoảng cách giữa mỗi điểm ảnh với giá trị centroids hiện tại và gán labels mỗi điểm với giá trị gần nhất.
- B5) Tính trung bình các điểm ảnh cùng labels và cập nhật lại centroids.
- B6) Lặp lại các bước 3-4-5 cho tới khi hết max_iters hoặc centroids không còn thay đổi
- B7) Khôi phục ảnh từ 1D (mảng 2D numpy) sang 2D (mảng 3D numpy).
- B8) Hiện thị ảnh và lưu ảnh theo định dạng file (png hoặc pdf).

MÔ TẢ CÁC HÀM

1. Hàm read_img

- **Chức năng:** Đọc hình ảnh từ đường dẫn file.
- **Đầu vào:** img_path: đường dẫn của file ảnh (string).
- **Đầu ra:** Trả về dưới dạng mảng 3D numpy (height, width, channels).
- **Mô tả:** Sử dụng module "Image" từ thư viện "PIL" gọi hàm open() truyền vào đường dẫn ảnh trả về đối tượng Image (một định dạng đặc biệt của "Pillow" cho phép thực hiện các thao tác khác nhau trên hình ảnh). Sau đó, gọi hàm array() từ thư viện "Numpy" chuyển đổi đối tượng Image thành mảng 3D (height, width, channels).

2. Hàm show_img

- **Chức năng:** Dùng để hiển thị hình ảnh 2D sử dụng thư viện matplotlib. Hàm này giúp kiểm tra kết quả của các bước xử lý ảnh.
- **Đầu vào:** img_2d: mảng "numpy" 3 chiều (height, width, channels) của hình ảnh.
- **Đầu ra:** Không trả về giá trị chỉ thực hiện chức năng hiển thị ảnh trên màn hình.
- **Mô tả:** Gọi các hàm hỗ trợ của thư viện matplotlib sau:
  - imshow(img_2d): hiển thị mảng 3D numpy được truyền vào (img_2d) dưới dạng hình ảnh trên không gian vẽ của matplotlib.
  - axis('off'): tắt các trục của hình ảnh để hiển thị đẹp mắt hơn.
  - show(): hiển thị hình ảnh đã được vẽ lên không gian vẽ của matplotlib ra giao diện "jupyter notebook".

3. Hàm save_img

- **Chức năng:** Lưu ảnh dưới định dạng PNG hoặc PDF. Nếu đường dẫn file không kết thúc bằng ".png" hoặc ".pdf", hàm sẽ thông báo lỗi.
- **Đầu vào:**
  - img_2d: Mảng 3D của ảnh.
  - img_path: đường dẫn lưu ảnh.
- **Đầu ra:** Không trả về giá trị chỉ thực hiện lưu ảnh vào file theo định dạng PNG hoặc PDF.
- **Mô tả:** Kiểm tra xem đường dẫn truyền vào có đuôi là "png" hay "pdf" hay không? Nếu không, sẽ in ra thông báo lựa chọn không hợp lệ. Ngược lại, sẽ tạo bộ nhớ hình ảnh từ một đối tượng mảng sau đó dùng hàm save() truyền vào đường dẫn để lưu với format là 3 ký tự cuối của đường dẫn, ngoài ra còn hiển thị hình ảnh tải về trên màn hình máy tính.

4. Hàm convert_img_to_1d

- **Chức năng:** Chuyển đổi ảnh từ dạng ma trận 2D (mảng 3D numpy) sang dạng ma trận 1D (mảng 2D numpy) để sử dụng trong quá trình xử lý hàm "kmeans".
- **Đầu vào:** img_2d: Mảng 3D numpy (height, width, channels) của ảnh.
- **Đầu ra:** Ảnh dưới dạng mảng 2D numpy (height * width, channels).
- **Mô tả:** Dùng hàm shape() trong thư viện "Numpy" để lấy kích thước ảnh gán vào 3 biến height, width, channels. Sau đó gọi hàm reshape() thay đổi hình dạng mảng thành 2D (height * width, channels) trong đó height * width = số điểm ảnh.

5. Hàm kmeans

- **Chức năng:** Hàm "kmeans" thực hiện thuật toán K-Means để phân cụm các điểm ảnh có sắc độ gần tương đồng.
- **Đầu vào:**
  - img_1d: Mảng 2D numpy của ảnh.
  - k_clusters: Số lượng cụm.
  - max_iter: Số lần lặp tối đa.
  - init_centroids: Phương pháp khởi tạo tâm cụm ('random' mặc định hoặc 'in_pixels').
- **Đầu ra:**
  - centroids: Mảng 2D numpy lưu trữ màu sắc của k cụm.
  - labels: Mảng 1D lưu trữ nhãn của từng điểm ảnh, xác định màu sắc thuộc về.
- **Mô tả:**
  - B1: Khởi tạo mảng zero "labels" với độ dài là số điểm ảnh và khởi tạo mảng "centroids" với các trường hợp "init_centroids" sau:
    - TH1 (init_centroids là random): Chạy vòng lặp, ngẫu nhiên giá trị trong khoảng từ 0 đến 255 với kích thước là ("k_clusters, 3 kênh màu "RGB") kiểu dữ liệu là float cho đến khi không có sự trùng lặp trong "centroids".
    - TH2 (init_centroids là in-pixels): Lấy các màu duy nhất trong ảnh và so sánh với "k_clusters". Nếu ít hơn "k_clusters" điều chỉnh lại "k_clusters" là giá trị số màu duy nhất. Sau đó chọn ngẫu nhiên "k_clusters" chỉ số điểm ảnh, khởi tạo "centroids" từ chỉ số điểm ảnh lấy được, chạy vòng lặp đảm bảo unique() giá trị màu duy nhất.
    - TH3: Thông báo sai giá trị "init_centroids" truyền vào khác "random" và "in_pixels".
  - B2: Sử dụng chuẩn Euclidean để tính khoảng cách từ mỗi điểm ảnh đến các giá trị "centroids" sau đó gán nhãn cho mỗi điểm ảnh với giá trị "centroids" gần nhất.
  - B3: Kiểm tra xem có giá trị nào mà không có điểm ảnh nào "labels" hay không? Nếu có chọn lại giá trị mới cho "centroid" đó để giải quyết trường hợp 3 "centroids" liên tiếp thì "centroid" ở giữa sẽ có điểm ảnh "label" tới là rỗng và không thể tính trung bình vì không thể chia cho 0. Tính trung bình các điểm ảnh trong mỗi giá trị "centroids" để tạo ra giá trị mới. Chuyển list giá trị "centroids" mới đó thành mảng "Numpy". Cập nhật mảng "centroids".
  - B4: Vòng lặp các B2+3+4 cho tới khi "max_iter" giảm về 0 hoặc khi "centroids" không còn cập nhật mới nữa. Trả về "centroids" và "labels".

6. Hàm generate_2d_img

- **Chức năng:** Khôi phục lại ảnh 2D từ "centroids" và "labels" sau khi chạy hàm "kmeans".
- **Đầu vào:**
  - img_2d_shape: Kích thước của ảnh gốc (height, width, channels).
  - centroids: Mảng 2D numpy lưu trữ các giá trị màu.
  - labels: Mảng 1D numpy lưu trữ nhãn của từng điểm ảnh đến các giá trị màu trong "centroids".
- **Đầu ra:** Ảnh dưới dạng mảng 3D (height, width, channels).
- **Mô tả:** Lấy kích thước ảnh gốc gán vào height, width và channels, gán màu trong "centroids" cho từng điểm ảnh được dán nhãn theo "labels" trả về từ hàm "kmeans" với kiểu "uint8" để có thể save ảnh. Sau đó chuyển từ mảng 2D (height * width, channels) sang 3D (height, width, channels) theo kích thước ảnh gốc và trả về dưới dạng mảng 3D.

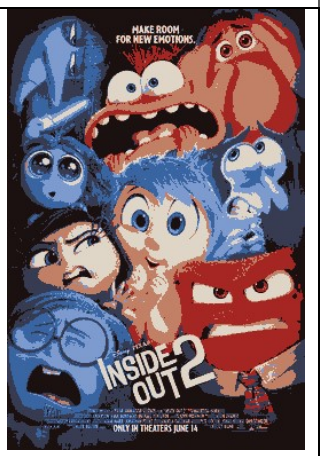
7. Hàm main

- **Chức năng:** gọi các hàm ở trên để thực hiện quá trình nén màu ảnh và hiển thị ra kết quả dưới các "k_clusters" khác nhau và "init_centroids" của cả "random" và "in-pixels" để phân tích và nhận xét.
- **Các bước thực hiện:**
  - B1: Nhập đường dẫn đến file ảnh trong máy, truyền vào hàm đọc ảnh (read_img) và chuyển ảnh đó sang mảng 2D bằng hàm "convert_img_to_1d" để xử lý trong hàm "kmeans".
  - B2: Để thuận tiện cho việc test code, khởi tạo mảng "k_clusters_list" = [3,5,7] và mảng "init_centroids_list" có các giá trị "random" và "in_pixels".
  - B3: Chạy 2 vòng lặp, gọi hàm "kmeans" trả về "centroids" và "labels" ứng với các giá trị khác nhau của "k_clusters_list" và "init_centroids_list". Sau đó, khôi phục ảnh bằng hàm "generate_2d_img" và hiển thị lên "jupyter notebook".
  - B4: Nhập đuôi file 1 trong 2 lựa chọn là "png" và "pdf" sau đó gộp với {k_clusters}_{init_centroids} thành đường dẫn truyền vào hàm "save_img" để lưu kết quả sau khi nén màu vào cùng folder với file "jupyter notebook".

KẾT QUẢ VÀ NHẬN XÉT

Hình ảnh poster Inside Out 2 (ảnh gốc ban đầu) (nguồn: IMDb)

Kết quả

K clusters / Init centroids	3	5	7
Random			

In pixels			

Hình với "k_clusters" = 16 random

Hình với "k_clusters" = 16 in-pixels

Nhận xét

- Hình ảnh ban đầu có 78465 màu sắc (kích thước 385px * 260px) và dung lượng là 36.9 KB. Trong khi các ảnh khác có số lượng màu và dung lượng như sau:

	Số lượng màu	Dung lượng ảnh (KB)
Random	3	15.6
	5	22.5
	7	27.8
In pixels	3	14.8
	5	22.5
	7	26.9

  ⇨ Có thể thấy với số lượng màu càng tăng thì dung lượng ảnh sẽ tăng nên việc nén màu ảnh giúp giảm dung lượng của ảnh.
- Với những ảnh có ít số lượng màu khi nén ảnh sẽ không giảm đáng kể chất lượng ảnh, tuy nhiên ở đây với poster của một phim chiếu rạp dành cho trẻ em thì số lượng màu khá lớn khi nén còn dưới 10 màu vẫn nhìn thấy các nhân vật nhưng màu sắc vốn có của các nhân vật sẽ thay đổi.
- Với "k_cluster" càng lớn thì hình ảnh sẽ rõ màu hơn (với "k_clusters" = 16 màu giữa các nhân vật đã rõ ràng) . Với các ảnh có số lượng màu sắc ít hơn khi nén màu ảnh sẽ không ảnh hưởng quá nhiều đến chất lượng ảnh.
- Với "random" thì kết quả không ổn định do sự khởi tạo ngẫu nhiên [0,255] ban đầu dẫn đến kết quả có thể thay đổi qua từng lần chạy code. Trong khi "in-pixel" khởi tạo dựa trên các pixel của ảnh nên kết quả ổn định và có xu hướng phân bố màu sắc tốt hơn [2][3].
- Tốc độ chạy của chương trình (khoảng 0.8-1s) tùy vào việc chọn ngẫu nhiên "centroids" ban đầu chưa tối ưu như việc triển khai "sklearn.cluster.Kmeans" (chỉ có 0.1-0.2s).
- Triển khai "sklearn.cluster.Kmeans" được tối ưu hóa cao và được sử dụng rộng rãi trong thực tế. Nó cung cấp phân cụm hiệu quả với các kỹ thuật khởi tạo mạnh như "k-means++", kết hợp lợi ích của cả hai phương pháp ngẫu nhiên và từ pixel bằng cách khởi tạo "centroids" thông minh để tăng tốc độ hội tụ và cải thiện chất lượng thuật toán[4].
- K-means: Tối thiểu hóa tổng bình phương khoảng cách giữa các điểm và tâm cụm tương ứng. Nó hiệu quả về mặt tính toán nhưng nhạy cảm với các giá trị ngoại lệ [5]. Nói chung, trung bình số học thực hiện điều này. Nó không tối ưu hóa khoảng cách mà là bình phương độ lệch so với giá trị trung bình [1].
- K-medians: Tối thiểu hóa tổng các độ lệch tuyệt đối từ trung vị, làm cho nó ít bị ảnh hưởng bởi các giá trị ngoại lệ nhưng phức tạp hơn về mặt tính toán [6]. Phương pháp này đặc biệt hữu ích cho các tập dữ liệu có phân phối không chuẩn hoặc có các giá trị ngoại lệ đáng kể. Nói chung, giá trị trung bình trên mỗi trục sẽ thực hiện điều này. Đây là một công cụ ước tính tốt cho giá trị trung bình, nếu muốn giảm thiểu tổng độ lệch tuyệt đối (tức là $\sum_i |x_i - y_i|$) , thay vì bình phương [1].

TÀI LIỆU THAM KHẢO

[1] *clustering - k-means vs k-median?* (2014, July 27). Cross Validated. Retrieved June 19, 2024, from https://stats.stackexchange.com/questions/109547/k-means-vs-k-median

[2] Saedsayad, "Clustering in Data Mining: K-means vs. K-medians," Data Mining Tutorials. Available at: Saedsayad Data Mining

[3] Understanding k-Medians: Definition, Explanations, Examples & Code, SERP AI. Available at: SERP AI

[4] Scikit-learn Documentation, "K-Means Clustering Algorithm." Available at: Scikit-learn Documentation

[5] Moshkovitz, M., & others. (2020). "A Discussion on Clustering Algorithms: K-means and K-medians." *International Conference on Machine Learning*. Available at: Proceedings of the 37th International Conference on Machine Learning

[6] "The complete guide to clustering analysis: k-means and hierarchical clustering," Stats and R. Available at: Stats and R

[7] *NumPy reference — NumPy v2.0 Manual*. (n.d.). NumPy -. Retrieved June 19, 2024, from https://numpy.org/doc/stable/reference/index.html

[8] Tài liệu lab 01: Vector & System of Linear Equations (Cô Uyên)

[9] guide, s. (n.d.). *Getting started*. Python Developer's Guide. Retrieved June 19, 2024, from https://devguide.python.org/getting-started/

[10] *Using Matplotlib — Matplotlib 3.9.0 documentation*. (n.d.). Matplotlib. Retrieved June 19, 2024, from https://matplotlib.org/stable/users/index

[11] *Jain, S. (2021, July 3). Python: Pillow (a fork of PIL).* GeeksforGeeks. Retrieved June 19, 2024, from https://www.geeksforgeeks.org/python-pillow-a-fork-of-pil/