TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN 1

COLOR COMPRESSION

Bộ môn : Toán ứng dụng và thống kê cho Công nghệ thông tin

Giảng viên lý thuyết: Vũ Quốc Hoàng

Giảng viên thực hành: Nguyễn Văn Quang Huy

Phan Thị Phương Uyên

Trần Thị Thảo Nhi

Mã số sinh viên : 20127206

Họ và tên : Vũ Đình Duy Khánh

Lớp : 20CLC05

I. Ý tưởng thực hiện :

Khi máy tính đọc vào một bức ảnh màu, máy tính sẽ nhận điện đó là một ma trận các điểm ảnh.

Với mỗi điểm ảnh là một vector [v1 , v2, v3], với v1, v2, v3 lần lượt tượng trưng cho các kênh màu Red, Green, Blue giá trị thuộc đoạn [0, 255]

Đặc điểm của thuật toán Kmeans Clustering là chúng ta không biết các điểm ảnh cho trước được gom cụm dựa vào tiêu chí nào, chỉ trả về số lượng cluster đưa vào. Do đó chúng ta có thể hoàn toàn chọn được số lượng cluster tùy ý để giảm số lượng màu mà chúng ta mong muốn.

Các bước xử lý ảnh :

- Nhận vào một bức ảnh, chuyển về ma trận phù hợp với tham số đầu vào của thuật toán Kmeans Clustering
- Xử lý ảnh với thuật toán Kmeans Clustering :
  + Khởi tạo k centroids : các centroids cũng là một vector có dạng [v1, v2, v3].
  + Phân cụm từng điểm ảnh dựa vào các centroids : gán cho các điểm ảnh giá trị index của centroid mà khoảng cách từ điểm ảnh đến centroid đó gần nhất.
  + Cập nhật các centroids : các centroids mới có giá trị bằng giá trị trung bình các cluster thuộc các centroids cũ.
  + Thực hiện lặp lại phân cụm và cập nhật centroids mới đến khi đạt kết quả hội tụ hoặc đạt đến giá trị max_inter.

Quy trình chạy chương trình :

- Nhập tên file ảnh, chọn số lượng cluster, chọn định dạng đầu ra.
- Xử lý ảnh.
- Xuất ra ảnh ban đầu và ảnh sau xử lý để so sánh.
- Lưu ảnh với định dạng đầu ra đã chọn.

II. Mô tả hàm:

1. Khởi tạo centroids :

```
1  def initCentroids(image, k_clusters, type):
2      if type == 'in_pixels':
3          return image[np.random.choice(image.shape[0], k_clusters, replace=False)]
4      elif type == 'random':
5          return np.random.choice(256, (k_clusters, image.shape[1]), replace=False)
6      else:
7          return None
```

Sử dụng hàm numpy.random.choice để trả về 1 kết quả (single item hoặc ndarray) từ mảng 1D truyền vào

Input :

- image (np.ndarray 1D) : ảnh đầu vào dưới dạng 1 chiều
- k_clusters (integer) : số clusters, k clusters trả về k centroids
- type (string) : ‘random’ hoặc ‘in_pixels’

Output :

- Với type = ‘in_pixels’ : trả về ngẫu nhiên k centroids là các vector điểm ảnh có trong ảnh đầu vào.
- Với type = ‘random’ : trả về ngẫu nhiên k centroids là các vector [v1, v2, v3] với mỗi giá trị là một số nguyên thuộc đoạn [0, 255].

2. Phân cụm :

```
def assignLabelToPixels(image, centroids):
    return np.argmin(np.linalg.norm(image - centroids[:, None], axis=2), axis=0)
```

Gán label là index cho các pixel gần đó nhất.

Dùng np.linalg.norm để tính khoảng cách Euclide từ điểm ảnh đến centroid, sau đó dùng np.argmin để trả về index của centroid có khoảng cách đến điểm ảnh nhỏ nhất.

Input :

- image (np.ndarray 1D) : ảnh đầu vào dưới dạng 1 chiều
- centroids (np.ndarray) : np.ndarray chứa các centroids

Output :

- np.ndarray chứa label của mỗi điểm ảnh, trong đó tại i là label của điểm ảnh thứ i trong ảnh.

3. Cập nhật các centroids :

```
def updateNewCentroids(image, labels, oldCentroids):
    centroids = np.zeros(oldCentroids)
    for i in range(oldCentroids[0]):
        pixels = image[labels == i]
        if pixels.shape[0]:
            centroids[i] = np.mean(pixels, axis=0)
    return centroids
```

Cập nhật lại các centroids mới từ giá trị trung bình của các điểm ảnh thuộc cluster của centroid cũ bằng hàm numpy.mean

Input :

- image (np.ndarray 1D) : ảnh đầu vào dưới dạng 1 chiều
- labels (np.ndarray) : chứa label của mỗi điểm ảnh
- oldCentroids (np.ndarray) : chứ các centroids cũ

Output :

- centroids (np.ndarray) : các centroids mới

4. Thuật toán K-means Clustering

```python
def KmeansClustering(image, k_clusters, max_iter = 1000, init_centroids = 'random'):
    centroids = initCentroids(image, k_clusters, init_centroids)
    labels = np.full(image.shape[0], -1)
    for i in range(max_iter):
        labels = assignLabelToPixels(image, centroids)
        old_centroids = centroids
        centroids = updateNewCentroids(image, labels, centroids.shape)
        # Kiểm tra đạt kết quả hội tụ chưa
        if np.allclose(old_centroids, centroids, rtol = 10e-3, equal_nan=False):
            break
    return centroids, labels

```

Thực hiện lặp lại việc phân cụm cluster và cập nhật centroids mới liên tục cho đến khi thỏa điều kiện dừng (đạt kết quả hội tụ) hoặc đạt max_iter.

np.allclose : trả về True nếu hai mảng có giá trị phần tử bằng nhau trong một dung sai rtol

Input :

- image (np.ndarray) : ảnh đầu vào dưới dạng 1 chiều
- k_clusters (integer) : số lượng cluster
- max_iter (integer) : số lần chạy thuật toán Kmeans tối đa
- init_centroids (string) : ‘random’ hoặc ‘in_pixels’

Output :

- centroids (np.ndarray) : các centroids sau khi chạy xong thuật toán
- labels (np.ndarray) : các label sau khi chạy xong thuật toán

5. Hàm main

```python
if __name__ == '__main__':
    filename = input('Nhập tên file ảnh : ')
    image = np.array(Image.open(filename))
    image_1d_array = image.reshape(image.shape[0] * image.shape[1], image.shape[2])
    kClusters = int(input('Nhập số lượng cluster : (Chỉ nhập số nguyên) '))
    #chọn đuôi file đầu ra, chọn 1 trong 2, nhập đến khi chọn đúng
    outputExtension = 0
    while (outputExtension != 1 and outputExtension != 2):
        outputExtension = int(input('Lựa chọn đầu ra : 1. png - 2. pdf : '))

    centroids, labels = KmeansClustering(image_1d_array, kClusters, max_iter = 1000, init_centroids='in_pixels')
    newImage = (centroids[labels].astype(np.uint8)).reshape(image.shape)

    Image.fromarray(newImage.astype(np.uint8)).save(filename.split('.')[0] + '-k' + str(kClusters) + ('.png' if outputExtension == 1 else '.pdf'))
```

- Khi chạy chương trình tiến hành nhập tên file, số lượng cluster, chọn đầu ra của ảnh (1. png / 2. pdf)
- Chuyển ảnh sang dạng ảnh 1 chiều (dòng 4)
- Thực hiện thuật toán Kmeans Clustering (dòng 11)
- Chuyển kết quả của thuật toán KmeansClustering lại thành ảnh với kích thước bằng kích thước ảnh gốc (dòng 12)
- Lưu file ảnh kết quả (dòng 13)

6. **Hiển thị ảnh lúc đầu và ảnh sau thực hiện để dễ dàng so sánh**

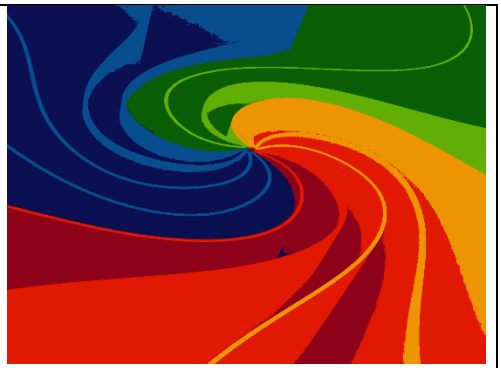
```
print('Ảnh lúc đầu : ')
plt.imshow(image)
```

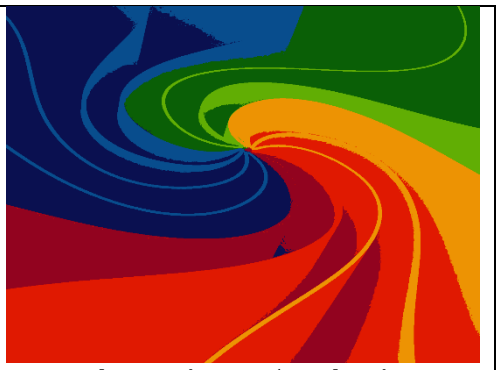
```
print('Ảnh sau giảm số lượng màu')
plt.imshow(newImage)
```

III. Hình ảnh kết quả :

1. **Ảnh gốc :**

2. **Ảnh với k = 3, 5, 7**

		
k = 3 với type = 'in_pixels'	k = 5 với type = 'in_pixels	k = 7 với type = 'in_pixels'

		
k = 3 với type = ‘random’	k = 5 với type = ‘random’	k = 7 với type ‘random’

3. Ảnh với k = 150, type = ‘in_pixels’

IV. Nhận xét kết quả

- Nhìn chung, chương trình cho kết quả khá tốt, có thể chấp nhận được.
- Với số lượng cluster càng lớn, hay số lượng màu lớn, hình ảnh càng được bảo toàn và rõ nét.
- Số lượng max_iter khá nhỏ để cho chương trình đưa ra kết quả tốt nhất.
- Bên cạnh đó, thời gian tính toán khá lâu khi có số lượng cluster lớn, tấm ảnh có số điểm ảnh lớn do chỉ sử dụng thư viện numpy nên việc tính toán trên ma trận kích thước lớn lâu hơn so với việc sử dụng các thư viện khác, đơn cử là scipy

V. Tài liệu tham khảo

NumPy

k-means clustering - Wikipedia

Machine Learning cơ bản (machinelearningcoban.com)

Machine Learning Tutorial Python - 13: K Means Clustering Algorithm - YouTube