

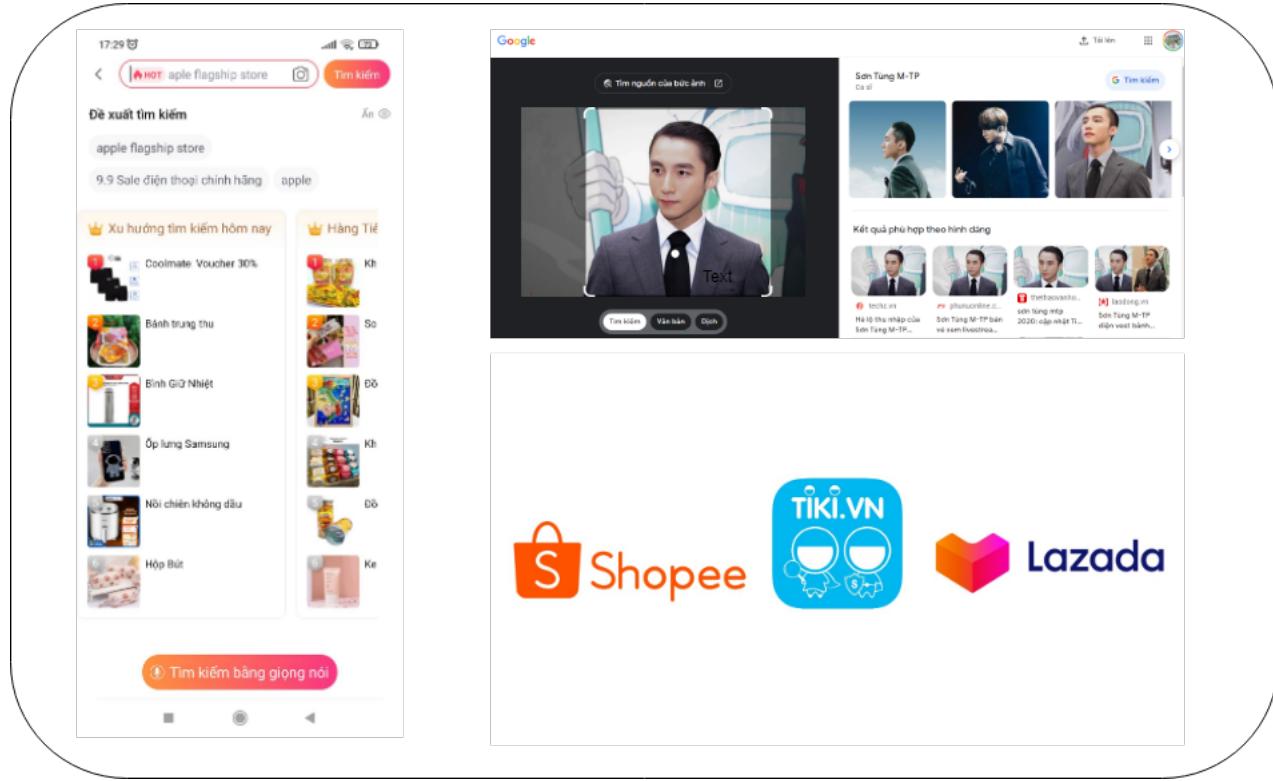
# AI VIET NAM – COURSE 2023

## Image Retrieval - Project

Ngày 20 tháng 8 năm 2023

### Phần I: Giới thiệu

**Image Retrieval (Tạm dịch: Truy vấn ảnh)** là một bài toán thuộc lĩnh vực Truy vấn thông tin (Information Retrieval). Trong đó, nhiệm vụ của chúng ta là xây dựng một chương trình trả về các hình ảnh (Images) có liên quan đến hình ảnh truy vấn đầu vào (Query), được lấy từ một bộ dữ liệu hình ảnh cho trước. Hiện nay, ta có thể điểm qua một số ứng dụng truy vấn ảnh như: Google Search Image, chức năng tìm kiếm sản phẩm bằng hình ảnh trên Shopee, Lazada, Tiki,...



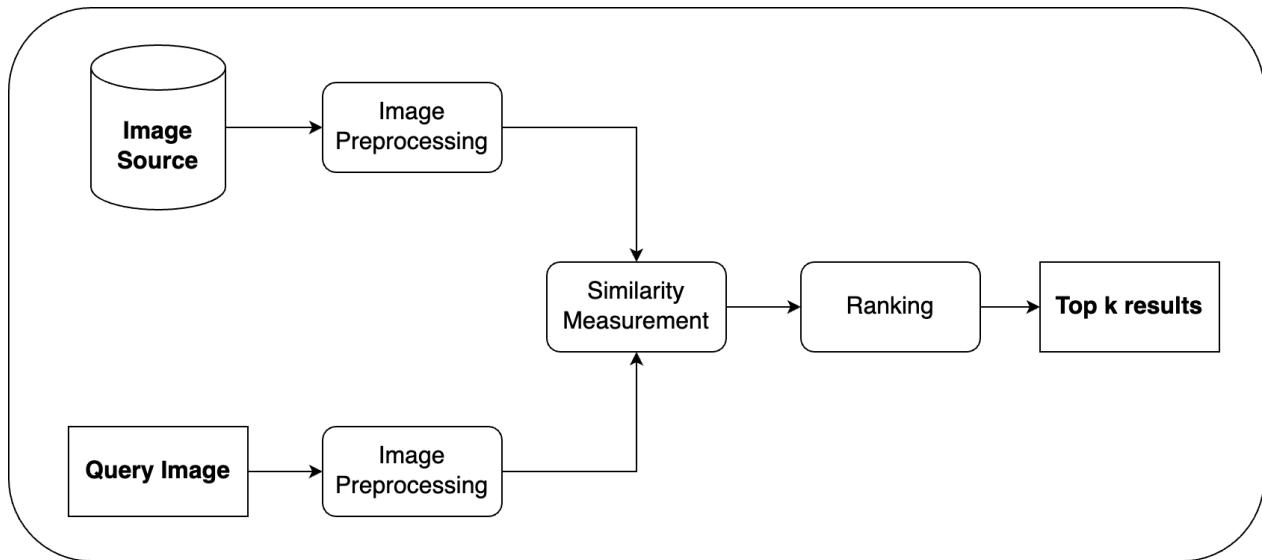
Trong project này, chúng ta sẽ xây dựng một chương trình Truy vấn ảnh có Input/Output như sau:

- **Input:** Ảnh truy vấn  $q$ ; Bộ ảnh  $I$ .
- **Output:** Danh sách các ảnh  $i$  ( $i \in I$ ) có liên quan đến ảnh truy vấn.

## Phần II: Cài đặt chương trình

### A. Xây dựng mô hình cơ bản (baseline)

Thông thường, một chương trình truy vấn thông tin thường được xây dựng theo pipeline như ảnh sau:



Hình 1: Pipeline tổng quát của một hệ thống truy vấn ảnh.

Theo đó, chúng ta sẽ thực hiện theo từng bước sau, code được xây dựng trên Google Colab:

1. **Tải bộ dữ liệu ảnh:** Để có dữ liệu nguồn (bộ ảnh **I**) cho hệ thống trả kết quả về, chúng ta cần phải có một bộ dữ liệu ảnh. Trong phần này, các bạn hãy tải về bộ dữ liệu ảnh có sẵn tại [đây](#).



Hình 2: Một vài mẫu ảnh trong bộ dữ liệu.

## 2. Import các thư viện cần thiết:

```

1 import os
2 import numpy as np
3 import cv2
4 import matplotlib.pyplot as plt

```

## 3. Đọc dữ liệu:

Sau khi đã giải nén file dữ liệu, chúng ta sẽ bắt đầu với việc đọc danh sách ảnh này lên code như sau:

```

1 dataset_dir = 'images_mr'
2 image_filenames = os.listdir(dataset_dir)
3 src_images = []
4 for filename in image_filenames:
5     filepath = os.path.join(
6         dataset_dir, filename
7     )
8     image = cv2.imread(filepath)
9     image = cv2.cvtColor(
10        image, cv2.COLOR_BGR2RGB
11    )
12    src_images.append(image)

```

Trong đó:

- **Dòng 1:** Khai báo đường dẫn đến thư mục chứa ảnh.
- **Dòng 2:** Sử dụng hàm `os.listdir()` để lấy danh sách các tên file ảnh.
- **Dòng 3:** Khai báo danh sách rỗng dùng để chứa ảnh đã đọc.
- **Dòng 4:** Duyệt qua từng tên ảnh.
- **Dòng 5, 6, 7:** Khai báo đường dẫn của ảnh đang xét bằng cách nối thư mục chứa ảnh và tên ảnh sử dụng hàm `os.path.join()`.
- **Dòng 8, 9, 10, 11:** Đọc ảnh và đổi kênh màu của ảnh từ BGR (mặc định của OpenCV) sang RGB.
- **Dòng 12:** Thêm ảnh vừa đọc được vào danh sách khai báo ở dòng 3.

## 4. Tiền xử lý dữ liệu ảnh:

Dể cho việc tính toán truy vấn ảnh nhanh và chính xác hơn, chúng ta sẽ áp dụng một số phương pháp tiền xử lý ảnh (bước tiền xử lý dữ liệu thường được áp dụng trong hầu hết các bài toán liên quan đến Machine Learning). Các kỹ thuật tiền xử lý ảnh được áp dụng trong chương trình bao gồm:

- **Resizing:** Thay đổi kích thước của ảnh về kích thước chung. Mặc định là (64, 64):

```

1 def image_resize(images, target_size=(64, 64)):
2     resized_image = cv2.resize(images, target_size)
3
4     return resized_image

```

- **Normalizing:** Chuẩn hóa giá trị trong ảnh sử dụng Z-score Normalization. Từ một ảnh  $X$  và danh sách ảnh  $I$ , công thức chuẩn hóa ảnh được diễn đạt như sau:

$$X_{normalized} = \frac{X - mean(I)}{std(I)}$$

Theo công thức, ta cần định nghĩa một hàm tính `mean` và `std` của danh sách ảnh:

```

1 def calculate_mean_std(images):
2     mean = np.mean(images, axis=(0, 1, 2))
3     std = np.std(images, axis=(0, 1, 2))
4
5     return mean, std

```

Sau đó, ta chỉ việc triển khai lại công thức thành một hàm normalize:

```

1 def image_std_normalize(images, mean, std):
2     normalized_image = (images - mean) / std
3
4     return normalized_image

```

- **Flattening:** Để có thể dễ dàng tính toán độ tương đồng giữa các cặp ảnh, ta sẽ chuyển ảnh thành vector 1D bằng phép flatten. Ở đây, ta thiết kế một hàm có thể xử lý hai trường hợp đầu vào: trường hợp chỉ có một ảnh và trường hợp có danh sách các ảnh.

```

1 def image_flatten(images, is_batch=False):
2     if is_batch:
3         flattened_image = images.reshape(images.shape[0], -1)
4     else:
5         flattened_image = images.reshape(-1)
6
7     return flattened_image

```

Theo đó, ta định nghĩa thêm một tham số `is_batch` để xác định kiểu đối tượng đầu vào.

Như vậy, kết hợp các phương thức tiền xử lý đã triển khai, chúng ta sẽ thiết kế hai hàm tiền xử lý như sau:

- **Hàm tiền xử lý cho danh sách ảnh:**

```

1 def preprocess_batches(images):
2     resized_images = [
3         image_resize(image) for image in src_images
4     ]
5     images_arr = np.array(resized_images)
6     mean, std = calculate_mean_std(images_arr)
7     normalized_images = image_std_normalize(
8         images_arr,
9         mean, std
10    )
11    flattened_images = image_flatten(
12        normalized_images,
13        is_batch=True
14    )
15
16    return flattened_images, mean, std

```

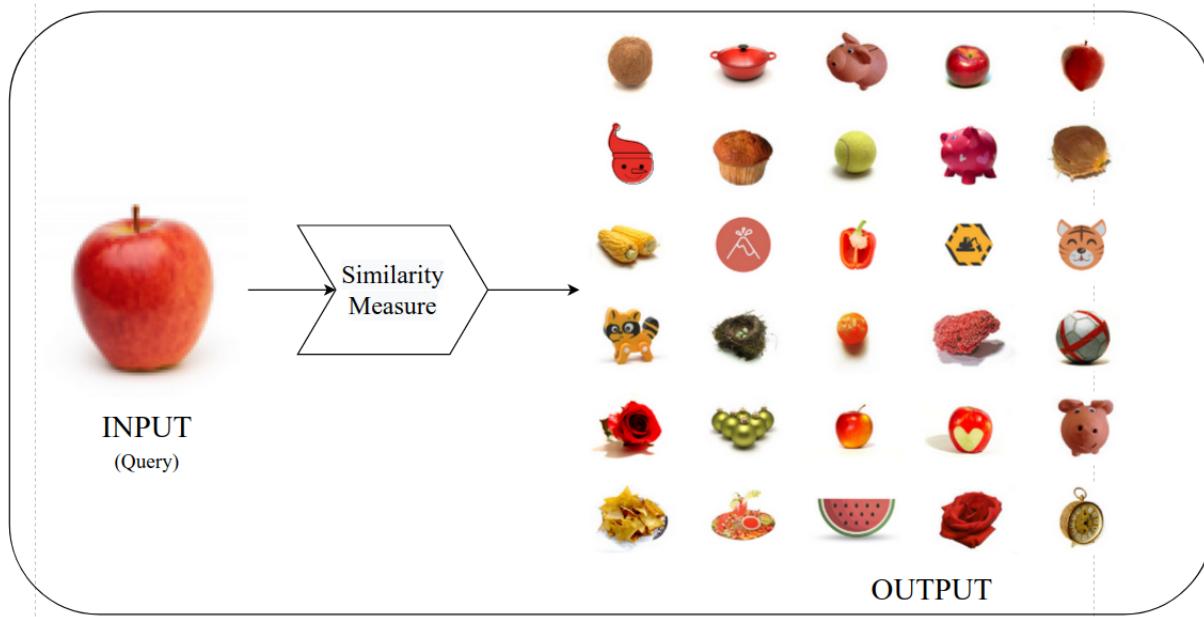
- **Hàm tiền xử lý cho ảnh truy vấn:**

```

1 def preprocess_query(image):
2     resized_image = image_resize(image)
3     normalized_image = image_std_normalize(
4         resized_image,
5         mean, std
6     )
7     flattened_image = image_flatten(normalized_image)
8
9     return flattened_image

```

5. **Xây dựng hàm tính độ tương đồng:** Để tìm được ảnh có nội dung liên quan đến ảnh truy vấn, ta có thể tận dụng các công thức tính độ tương đồng giữa hai vector, sử dụng kết quả của độ đo như là điểm tương đồng giữa các cặp ảnh.



Hình 3: Sử dụng các hàm tính độ tương đồng giữa hai vector để tìm giá trị tương đồng giữa ảnh truy vấn và các ảnh trong bộ dữ liệu.

Với vector ảnh truy vấn  $q$  và vector ảnh  $c$ , cả hai vector đều có kích thước  $n$ , các độ đo mà chúng ta sử dụng được định nghĩa như sau:

- **Mean Absolute Error (MAE):**

$$MAE(q, c) = \frac{1}{n} \sum_{i=1}^n |q_i - c_i|$$

- **Mean Squared Error (MSE):**

$$MSE(q, c) = \frac{1}{n} \sum_{i=1}^n (q_i - c_i)^2$$

- **Cosine Similarity:**

$$\text{cosine\_similarity}(q, c) = \frac{q \cdot c}{\|q\| \|c\|}$$

- **Correlation Coefficient:**

$$\text{correlation\_coefficient}(q, c) = \frac{n (\sum_{i=1}^n q_i \cdot c_i) - (\sum_{i=1}^n q_i) (\sum_{i=1}^n c_i)}{\sqrt{n \sum_{i=1}^n q_i^2 - (\sum_{i=1}^n q_i)^2} \sqrt{n \sum_{i=1}^n c_i^2 - (\sum_{i=1}^n c_i)^2}}$$

Theo định nghĩa của các công thức trên, chúng ta sẽ triển khai code để tính độ tương đồng giữa vector câu truy vấn và danh sách các vector ảnh của bộ dữ liệu như sau:

```

1 def mean_absolute_error(query_vector, src_vectors):
2     abs_diff = np.abs(src_vectors - query_vector)
3     mae = np.mean(abs_diff, axis=1)
4
5     return mae
6
7 def mean_squared_error(query_vector, src_vectors):
8     squared_diff = (src_vectors - query_vector) ** 2
9     mse = np.mean(squared_diff, axis=1)
10
11    return mse
12
13 def cosine_similarity(query_vector, src_vectors):
14     query_norm = np.linalg.norm(query_vector)
15     normalized_query = query_vector / query_norm
16     src_norms = np.linalg.norm(src_vectors, axis=1)
17     normalized_src = src_vectors / src_norms[:, np.newaxis]
18
19     cosine_similarity = np.dot(normalized_src, normalized_query)
20
21    return cosine_similarity
22
23 def correlation_coefficient(query_vector, src_vectors):
24     return np.corrcoef(query_vector, src_vectors)[-1, -1]

```

6. **Xây dựng hàm xếp hạng:** Với tất cả các code thành phần đã triển khai, chúng ta sẽ kết hợp lại để tạo thành hàm trả về kết quả xếp hạng độ tương đồng giữa ảnh truy vấn và danh sách ảnh nguồn như sau:

```

1 def ranking(preprocessed_query_image, preprocessed_src_images, top_k=10):
2     scores = cosine_similarity(
3         preprocessed_query_image,
4         preprocessed_src_images
5     )
6     ranked_list = np.argsort(scores)[-1][:-top_k]
7     scores = scores[ranked_list]
8
9     return ranked_list, scores

```

Trong đó:

- **Dòng 1:** Khai báo hàm `ranking()` với 3 tham số đầu vào gồm: array ảnh truy vấn (`query_image`), danh sách vector ảnh nguồn (`src_images`) và số lượng kết quả trả về (`top_k`).
- **Dòng 2, 3, 4, 5:** Thực hiện tính độ tương đồng giữa vector ảnh truy vấn so với các vector ảnh nguồn. Ở đây ta dùng `cosine_similarity()`, các bạn có thể thay thế bằng những hàm tính độ tương đồng khác.
- **Dòng 6:** Sắp xếp điểm tương đồng theo thứ tự giảm dần về mặt giá trị, biểu diễn kết quả theo chỉ mục (arg), đồng thời chỉ lấy `top_k` kết quả.
- **Dòng 7:** Lấy điểm tương đồng ứng với `top_k` kết quả.
- **Dòng 9:** Trả về danh sách xếp hạng và điểm tương đồng.

7. **Thực hiện truy vấn:** Với toàn bộ các thành phần đã xây dựng được, ta tiến hành xây dựng một đoạn code nhỏ cho phép truy vấn với ảnh bất kỳ. Thực hiện như sau:

```

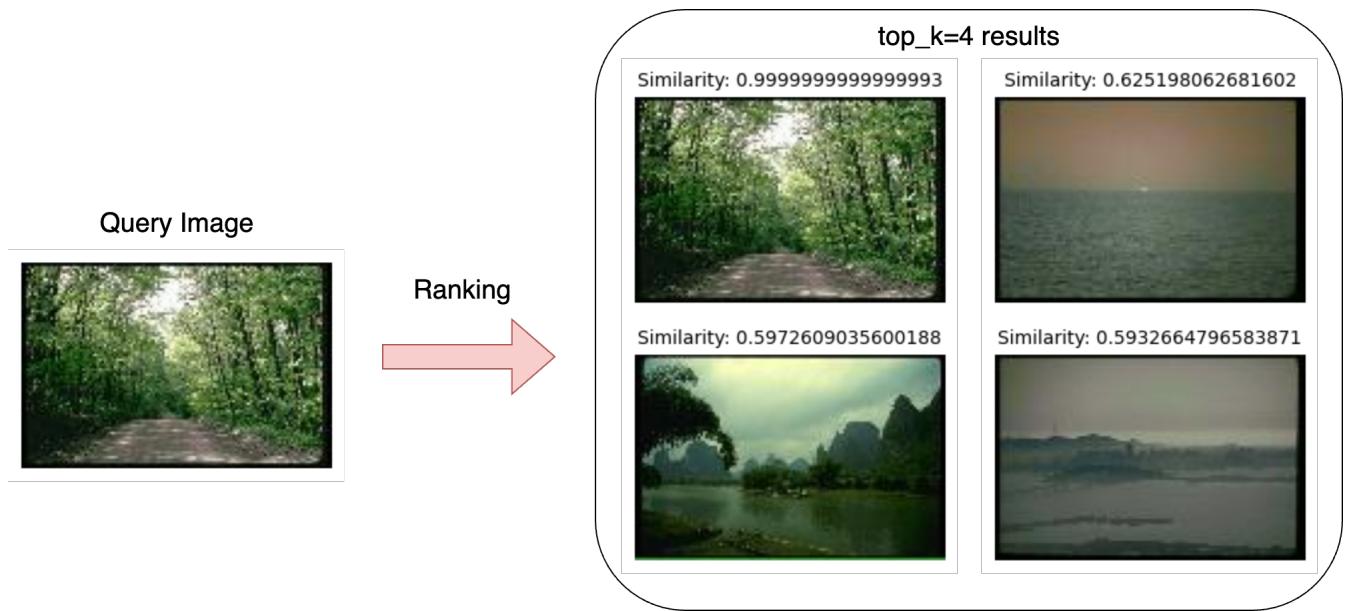
1 query_image_paths = [
2     '/content/images_mr/1885.jpg'
3 ]

```

```
4 top_k = 10
5
6 for query_image_path in query_image_paths:
7     query_image = cv2.imread(query_image_path, 1)
8     query_image = cv2.cvtColor(query_image, cv2.COLOR_BGR2RGB)
9     preprocessed_query_image = preprocess_query(query_image)
10
11     ranked_list, scores = ranking(
12         preprocessed_query_image,
13         preprocessed_src_images,
14         top_k
15     )
16
17     print('Query Image')
18     plt.figure(figsize=(3, 3))
19     plt.imshow(query_image)
20     plt.axis('off')
21     plt.show()
22     print(f'Top {top_k} results')
23     for idx in range(len(ranked_list)):
24         src_image_idx = ranked_list[idx]
25         similarity_score = scores[idx]
26         plt.figure(figsize=(3, 3))
27         plt.imshow(src_images[src_image_idx])
28         plt.title(f'Similarity: {similarity_score}', fontsize=10)
29         plt.axis('off')
30         plt.show()
```

Trong đó:

- **Dòng 1, 2, 3:** Khai báo danh sách chứa đường dẫn đến ảnh truy vấn. Ở đây ta lấy ví dụ một ảnh trong danh sách ảnh nguồn.
- **Dòng 4:** Khai báo số lượng kết quả trả về.
- **Dòng 6, 7, 8, 9:** Duyệt qua từng ảnh truy vấn, thực hiện đọc, đổi kênh màu và tiền xử lý ảnh.
- **Dòng 11, 12, 13, 14, 15:** Thực hiện truy vấn.
- **Dòng 18:** Cài đặt kích thước khung ảnh là (3, 3).
- **Dòng 19:** Cài đặt hiển thị ảnh dùng hàm `imshow()`.
- **Dòng 20:** Tắt trực hiển thị giá trị tọa độ x, y.
- **Dòng 21:** Hiển thị toàn bộ ảnh đã cài đặt vào matplotlib (ở đây là ảnh ở dòng 19).
- **Từ dòng 23 đến dòng 30:** Thực hiện tương tự việc hiển thị ảnh cho từng kết quả truy vấn sử dụng matplotlib.

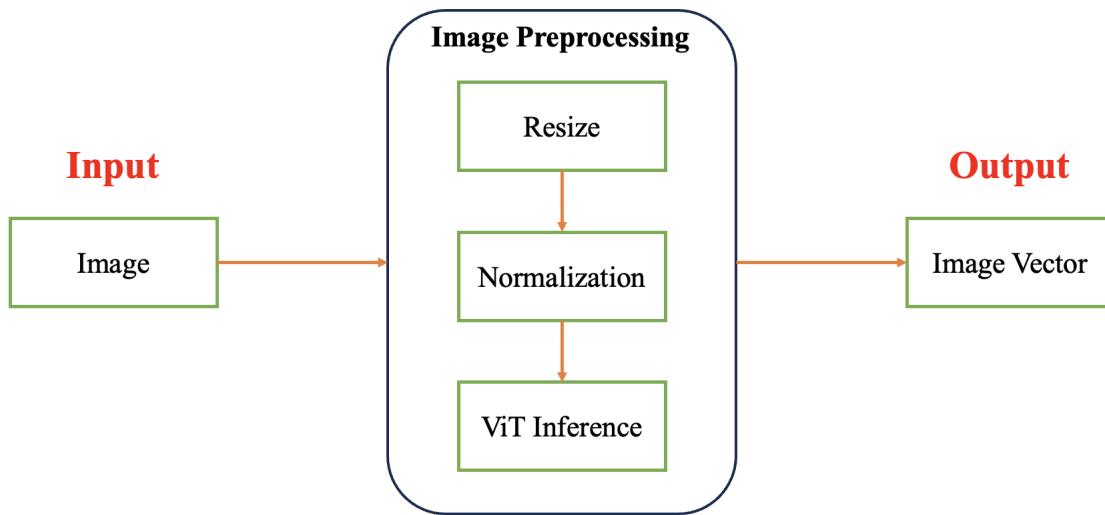


Hình 4: Minh họa kết quả truy vấn với  $top\_k = 4$ . Với việc ảnh truy vấn nằm trong danh sách ảnh nguồn, hiển nhiên ảnh có điểm tương đồng cao nhất là chính nó.

Như vậy, chúng ta đã hoàn tất xây dựng một hệ thống truy vấn ảnh. Với kết quả đạt được, ta có thể nhận thấy một cách trực quan rằng hệ thống hiện tại (baseline) chưa đạt được hiệu năng như mong muốn. Chính vì vậy, ta cần có những cải tiến để việc truy vấn trở nên tốt hơn.

## B. Truy vấn ảnh với đặc trưng từ pre-trained Vision Transformer (ViT)

Có rất nhiều cách cải thiện kết quả truy vấn mà ta có thể thử nghiệm. Trong đó, sử dụng những mô hình deep learning đã được huấn luyện trên một bộ dữ liệu cực lớn (pre-trained models) để tạo một dạng biểu diễn mới cho vector ảnh sẽ mang lại hiệu quả rất cao.



Hình 5: Sử dụng mô hình ViT để trích xuất đặc trưng ảnh đầu vào, giúp cho vector biểu diễn trở nên tốt hơn.

Trong phần này, chúng ta sẽ tìm hiểu cách áp dụng mô hình Vision Transformer (ViT), một trong những kiến trúc có hiệu năng rất mạnh mẽ ở thời điểm hiện tại, nhằm cải thiện hệ thống truy vấn ảnh. Mô hình được giới thiệu trong paper [An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale](#). Dựa trên chương trình đã xây dựng ở phần trên, ta sẽ tích hợp ViT theo các bước như sau (**Lưu ý:** các bạn cần kích hoạt GPU cho Google Colab):

- Import các thư viện cần thiết:** Để sử dụng được pre-trained ViT, ta sẽ tải thư viện transformers:

```
1 !pip install transformers==4.31.0 -q
```

Sau đó, import các thư viện sẽ sử dụng trong bài:

```
1 import os
2 import numpy as np
3 import cv2
4 import matplotlib.pyplot as plt
5 import torch
6 from transformers import ViTImageProcessor, ViTForImageClassification
```

- Đọc dữ liệu:** Các bước đọc dữ liệu sẽ tương tự như code baseline, song các bạn lưu ý nếu sử dụng colab thường sẽ không thể xử lý được toàn bộ dataset cho giới hạn phần cứng. Vì vậy, các bạn nên giới hạn ảnh nguồn khoảng 500 ảnh cho tiện việc test chương trình (có thể mở rộng hơn nếu phần cứng cho phép):

```
1 dataset_dir = 'images_mr'
2 image_filenames = os.listdir(dataset_dir)[:500]
3 src_images = []
```

```

4 for filename in image_filenames:
5     filepath = os.path.join(
6         dataset_dir,
7         filename
8     )
9
10    image = cv2.imread(filepath)
11    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
12    src_images.append(image)

```

**3. Khai báo mô hình ViT:** Để sử dụng được mô hình ViT, chúng ta cần khai báo chúng như sau:

```

1 device = 'cuda' if torch.cuda.is_available() else 'cpu'
2 processor = ViTImageProcessor.from_pretrained("google/vit-base-patch16-224")
3 model = ViTForImageClassification.from_pretrained("google/vit-base-patch16-224").
        to(device)

```

Trong đó:

- **Dòng 1:** Kiểm tra xem GPU có được kích hoạt hay không. Nếu không sẽ sử dụng CPU.
- **Dòng 2:** Khai báo ViTImageProcessor, dùng để chuẩn hóa ảnh đầu vào trước khi đưa vào mô hình ViT.
- **Dòng 3:** Khai báo mô hình ViT.

**4. Xây dựng hàm tiền xử lý ảnh:** Tận dụng code đã có, chúng ta sẽ tích hợp các biến đã khai báo ở trên để hoàn tất hàm trích xuất đặc trưng ảnh dùng ViT:

```

1 def preprocessing(images):
2     inputs = processor(
3         images,
4         return_tensors='pt',
5     ).to(device)
6
7     with torch.no_grad():
8         output = model(
9             **inputs,
10            output_hidden_states=True
11        ).hidden_states[-1][:, 0, :].detach().cpu().numpy()
12
13     return output

```

Trong đó:

- **Dòng 1:** Khai báo hàm `preprocessing()` nhận tham số đầu vào là ảnh (hoặc danh sách ảnh).
- **Dòng 2, 3, 4, 5:** Thực hiện chuẩn hóa ảnh đầu vào.
- **Dòng 7, 8, 9, 10, 11:** Dưa ảnh đã chuẩn hóa vào mô hình ViT để trích xuất đặc trưng.
- **Dòng 13:** Trả về đặc trưng ảnh trích được từ mô hình ViT.

Sau đó, ta tiến hành tiền xử lý kho ảnh nguồn:

```
1 preprocessed_src_images = preprocessing(src_images)
```

**5. Xây dựng hàm tính độ tương đồng:** Chúng ta sẽ sử dụng lại các hàm đã xây dựng ở baseline, ta sẽ chọn một hàm tiêu biểu là Cosine Similarity:

```

1 def cosine_similarity(query_vector, src_vectors):
2     query_norm = np.linalg.norm(query_vector)
3     normalized_query = query_vector / query_norm
4     src_norms = np.linalg.norm(src_vectors, axis=1)
5     normalized_src = src_vectors / src_norms[:, np.newaxis]
6
7     cosine_similarity = np.dot(normalized_src, normalized_query)
8
9     return cosine_similarity

```

6. **Xây dựng hàm xếp hạng:** Tương tự với hàm tính điểm tương đồng, hàm xếp hạng cũng sẽ giữ nguyên tương tự như ở baseline:

```

1 def ranking(preprocessed_query_image, preprocessed_src_images, top_k=10):
2     scores = cosine_similarity(
3         preprocessed_query_image,
4         preprocessed_src_images
5     )
6     ranked_list = np.argsort(scores)[::-1][:top_k]
7     scores = scores[ranked_list]
8
9     return ranked_list, scores

```

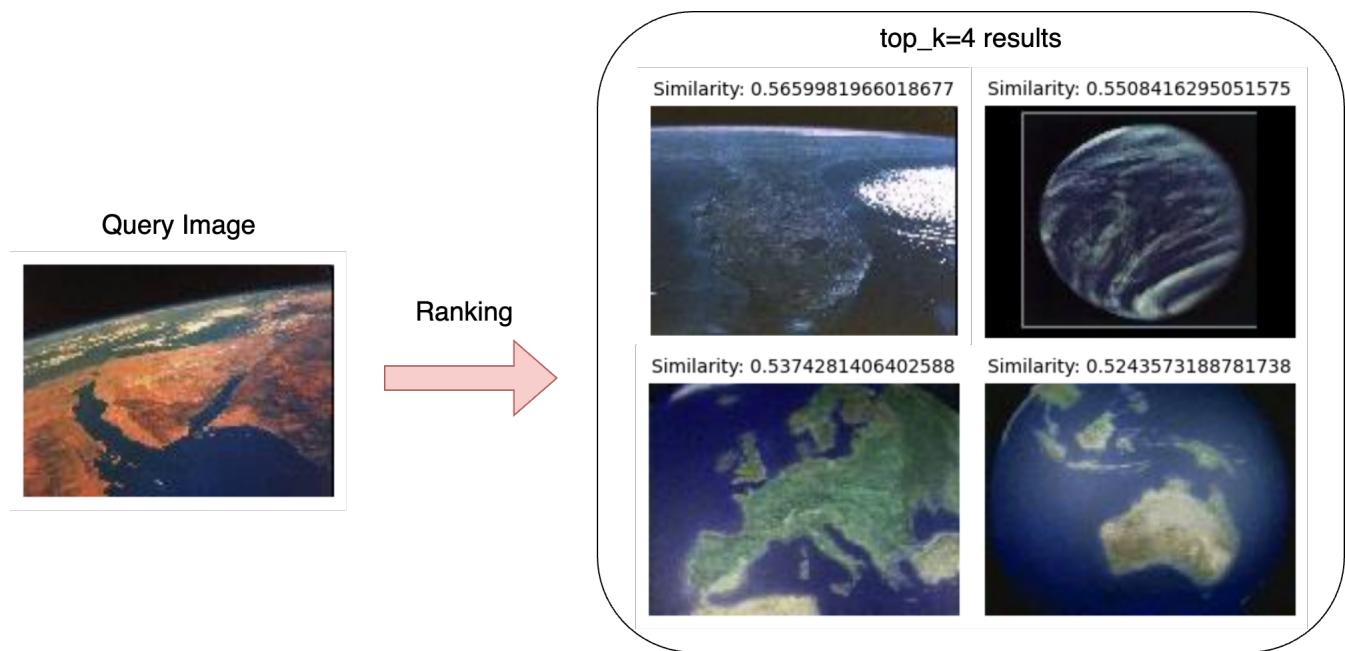
7. **Thực hiện truy vấn:** Cuối cùng, chúng ta thực hiện truy vấn với code cài đặt như sau:

```

1 query_image_paths = [
2     '/content/images_mr/615.jpg'
3 ]
4 top_k = 10
5
6 for query_image_path in query_image_paths:
7     query_image = cv2.imread(query_image_path, 1)
8     query_image = cv2.cvtColor(query_image, cv2.COLOR_BGR2RGB)
9     preprocessed_query_image = preprocessing(query_image).squeeze(0)
10
11     ranked_list, scores = ranking(
12         preprocessed_query_image,
13         preprocessed_src_images,
14         top_k
15     )
16
17     print('Query Image')
18     plt.figure(figsize=(3, 3))
19     plt.imshow(query_image)
20     plt.axis('off')
21     plt.show()
22     print(f'Top {top_k} results')
23     for idx in range(len(ranked_list)):
24         src_image_idx = ranked_list[idx]
25         similarity_score = scores[idx]
26         plt.figure(figsize=(3, 3))
27         plt.imshow(src_images[src_image_idx])
28         plt.title(f'Similarity: {similarity_score}', fontsize=10)
29         plt.axis('off')
30         plt.show()

```

Dễ dàng nhận thấy, kết quả truy vấn trả về đã cải thiện rõ rệt một cách trực quan.



Hình 6: Minh họa kết quả truy vấn với  $top\_k = 4$  sử dụng đặc trưng từ ViT.

Quan sát kĩ, chúng ta có thể thấy các đối tượng trả về tuy không hoàn toàn giống với ảnh truy vấn, song về tính chất cũng như ngữ cảnh lại có liên quan rất cao đến dữ liệu đầu vào.

## Phần III: Trắc nghiệm

1. Trong Image Retrieval, format của tham số truy vấn đầu vào là gì?
 

(a) File văn bản. (b) File ảnh.	(c) File âm thanh. (d) Bất kì file nào.
------------------------------------	--
2. Thư viện/module nào dưới đây có hỗ trợ hàm liệt kê danh sách tên các file ảnh một folder?
 

(a) os (b) numpy	(c) cv2 (d) matplotlib
---------------------	---------------------------
3. Để đọc một ảnh từ đường dẫn cho trước, ta sử dụng hàm nào dưới đây?
 

(a) cv2.imshow() (b) cv2.imwrite()	(c) cv2.imencode() (d) cv2.imread()
---------------------------------------	--
4. Đoạn code nào sau đây có thể thay đổi kênh màu của ảnh `image` từ BGR sang RGB?
 

(a) cv2.cvtColor(image, cv2.COLOR_RGB2BGR) (b) cv2.cvtColor(image, cv2.COLOR_BGR2HSV)	(c) cv2.cvtColor(image, cv2.COLOR_BGR2RGB) (d) cv2.cvtColor(image, cv2.COLOR_RGB2HSV)
--	--
5. Đoạn code nào sau đây đưa ảnh `image` về miền giá trị nhỏ hơn ban đầu?
 

(a) image = cv2.resize(image, (64, 64)) (b) image = image.flatten()	(c) image = (image - mean) / std (d) image = image + 255
--	---
6. Đoạn code nào sau đây làm thay đổi chiều dài và chiều rộng của ảnh `image` sang (64, 64)?
 

(a) image = cv2.resize(image, (64, 64)) (b) image = image.flatten()	(c) image = (image - mean) / std (d) image = image + 255
--	---
7. Bước nào sau đây không nằm trong các phương pháp áp dụng cho hàm tiền xử lý ảnh?
 

(a) Thay đổi kích thước ảnh. (b) Chuẩn hóa ảnh.	(c) Flatten ảnh. (d) Trực quan hóa ảnh.
--	--
8. Cho ma trận  $A = \begin{pmatrix} 4 & 7 \\ 9 & 3 \end{pmatrix}$ . Khi thực hiện phép flatten, ma trận A trở thành?
 

(a) (9, 3, 4, 7) (b) (4, 7, 9, 3)	(c) (9, 3, 7, 4) (d) (4, 7, 3, 9)
--------------------------------------	--------------------------------------
9. Cho một biến python `scores = [0.4, 0.2, 0.1, 0.7, 0.9]`. Khi thực hiện `scores = scores[::-1]`, giá trị của `scores` trở thành?

- (a) [0.4, 0.2, 0.1, 0.7, 0.9]  
(b) [0.1, 0.2, 0.4, 0.7, 0.9]

- (c) [0.9, 0.7, 0.1, 0.2, 0.4]  
(d) [0.9, 0.7, 0.4, 0.2, 0.1]

10. Để hiển thị một ảnh màu sử dụng matplotlib, ta dùng hàm nào sau đây?

- (a) plt.figure()  
(b) plt.plot()

- (c) plt.imshow()  
(d) plt.show()

11. Cho bộ ảnh  $Q, W, E, R$  với giá trị cosine similarity so với ảnh  $X$  lần lượt là 0.44, 0.12, 0.75, 0.56. Khi đó, ảnh có độ tương đồng cao thứ 2 so với ảnh  $X$  là?

- (a)  $Q$   
(b)  $W$

- (c)  $E$   
(d)  $R$

12. Cho một array  $a = np.array([[1, 2, 3], [4, 5, 6]])$ . Khi thực hiện  $a = a[..., np.newaxis]$ , shape của array này trở thành?

- (a) (2, 3, 1)  
(b) (2, 3)

- (c) (1, 2, 3)  
(d) (3, 2)

13. Trong phần code ViT, khi thực thi `output = model(**inputs, output_hidden_states=True).hidden_states`, số lượng phần tử có trong `output` là?

- (a) 10  
(b) 11

- (c) 12  
(d) 13

14. Trong phần code ViT, khi thực thi `output = model(**inputs, output_hidden_states=True).hidden_states[-1]`. Với  $\text{len}(\text{inputs}) = 1$ , shape của `output` có dạng?

- (a) (1, 768, 197)  
(b) (1, 197, 768)

- (c) (768, 197, 1)  
(d) (197, 768, 1)

15. Lợi ích lớn nhất trong việc sử dụng kết quả từ pre-trained ViT làm vector biểu diễn ảnh so cách thực hiện ở baseline là gì?

- (a) Tốc độ truy vấn nhanh.  
(b) Sử dụng ít tài nguyên tính toán.

- (c) Khả năng biểu diễn ảnh tốt hơn.  
(d) Thỏa mãn được information need.

- *Hết* -