Name: Nayan Biramane

Roll No: 21

Aim: To Create Program to perform a retrieving Image and Searching

Objective:
The fundamental need of any image retrieval model is to search and arrange the images that are in a visual semantic relationship with the query given by the user. Most of the search engines on the Internet retrieve the images based on text-based approaches that require captions as input.

Theory:

The technique of finding and arranging photographs so that their visual content is reflected is known as image retrieval. Content-based image retrieval (CBIR) works with the visual characteristics of images themselves, as opposed to text-based image retrieval, which depends on textual metadata like captions or keywords. This section will examine the basic ideas and methods that form the basis of a CBIR system.

1. Feature extraction: Images in CBIR are represented by a collection of features that encapsulate their visual attributes. These features can be more complex, like deep learning based feature vectors, or simpler, like color histograms and texture descriptors.In contemporary CBIR systems, Convolutional Neural Networks (CNNs) are frequently utilized for feature extraction. CNN models with pre-training, such as VGG, ResNet, or Inception, are capable of converting pictures into high-dimensional feature vectors.The goal of feature extraction is to produce a concise and insightful depiction of the visual information of the image. Efficient comparison and retrieval are made possible by this model.

2. Similarity Metrics: A similarity measure is used to compare the feature vectors of the query image and database photos in order to find related images.Depending on the type of characteristics being employed, common similarity measures include cosine similarity, Euclidean distance, or Jaccard similarity.Since it calculates the cosine of the angle between the vectors and provides a measure of their similarity independent of vector length, cosine similarity is frequently chosen for feature vectors.

3. Query Processing: The same technique used to extract features from database photos is applied when a user submits a query image.Next, using the selected similarity metric, these query attributes are compared to the features of photos stored in the database.

4. Ranking and Retrieval: A list of database photos ranked according to how similar they are to the query image is the outcome of the similarity comparison.An ordered retrieval result is displayed at the top of the list with the images most comparable to the query.

5. Difficulties: Image variability: It might be difficult to create reliable feature representations due to differences in scale, viewpoint, lighting, and backdrop in images. Scalability: Managing extensive image repositories effectively is a major obstacle. In large-scale systems, indexing and retrieval speed become crucial. Semantic gap: The accuracy of retrieval may be impacted by a mismatch between high-level semantic content in images and low-level visual aspects.
6. Enhancements:
Fusion of several features: Retrieval speed can be improved by integrating the findings of different feature types. Permitting users to offer input on the photographs they have retrieved in order to improve future searches in terms of relevance. The application of machine learning models to enhance ranking algorithms and acquire feature embeddings is known as machine learning techniques.

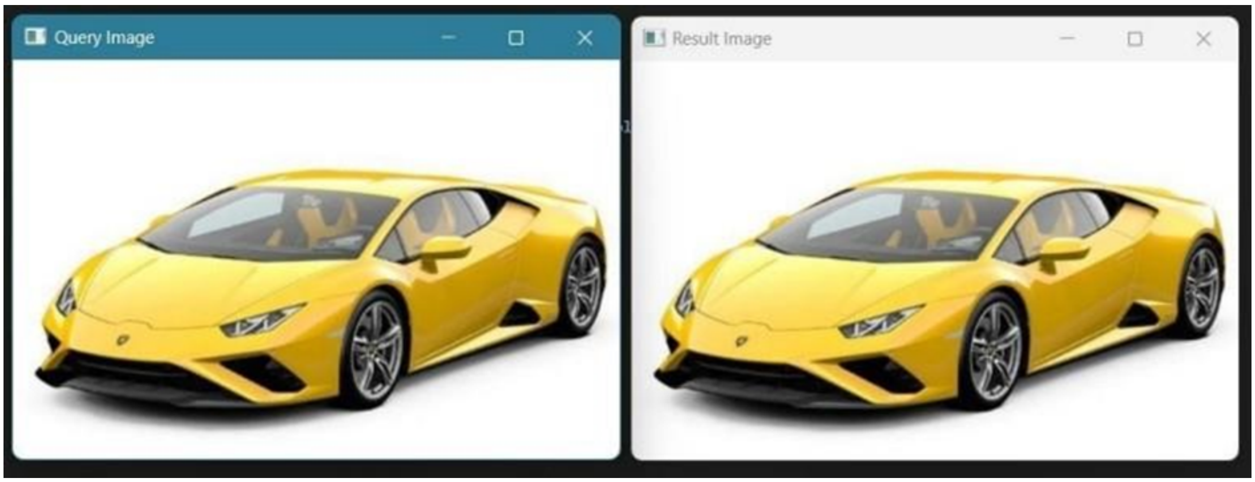
Code:

```
import cv2 import
numpy as np
from sklearn.metrics.pairwise import
cosine_similarity import os def
extract_features(image_path): image =
cv2.imread(image_path)
   hist = cv2.calcHist([image], [0, 1, 2], None, [8, 8, 8], [0, 256, 0,
   256, 0, 256]) hist = cv2.normalize(hist, hist).flatten() return hist
# Function to search for a query image in the images
folder def search_image(query_image_path,
images_folder): query_features =
extract_features(query_image_path) image_paths =
[] similarities = [] for root, dirs, files in
os.walk(images_folder):
    for file in files: if file.endswith(('.jpg',
     '.jpeg', '.png', '.bmp')): image_path =
     os.path.join(root, file) image_features =
     extract_features(image_path)
     similarity = cosine_similarity([query_features], [image_features])
     image_paths.append(image_path)
     similarities.append(similarity)
```

```
most_similar_idx = np.argmax(similarities)
most_similar_image_path =
image_paths[most_similar_idx] return
most_similar_image_path

if name == ' main ':
  images_folder = 'images'
  query_image_path =
  '/content/image.png'
  result_image_path = search_image(query_image_path,
  images_folder) if result_image_path:
    result_image = cv2.imread(result_image_path)
  cv2.imshow('Query Image',
  cv2.imread(query_image_path)) cv2.imshow('Result
  Image', result_image) cv2.waitKey(0)
  cv2.destroyAllWindows() else:
   print('No matching image found.')
```

Output:



Conclusion:

The experiment highlighted the importance of efficient image retrieval systems in various applications. Image retrieval and searching plays a vital role in applications. This experiment showcased the potential for practical applications in fields such as art recognition, content-based image retrieval, and visual search engines. As technology continues to advance, further exploration and innovation in this domain will undoubtedly lead to more efficient solutions