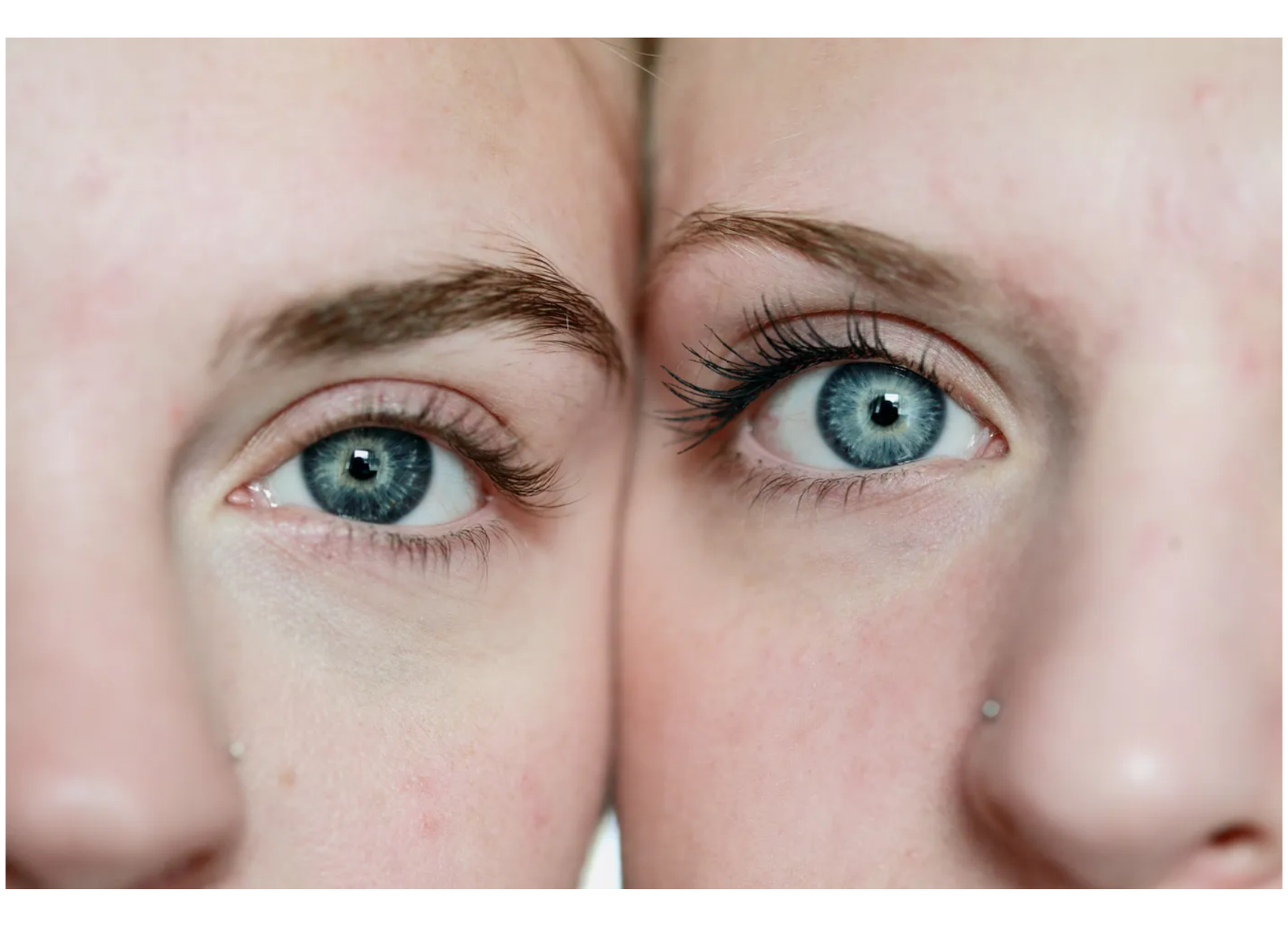
Exploring Image Similarity Approaches in Python

Vasista Reddy · Follow
Published in ScrapeHero · 5 min read · Sep 5, 2023



In a world inundated with images, the ability to measure and quantify the similarity between images has become a critical task. Whether it's for image retrieval, content recommendation, or visual search, image similarity approaches play a pivotal role in modern applications.

Fortunately, Python offers a plethora of tools and libraries that make exploring and implementing these approaches accessible to developers and researchers alike. In this blog, we'll delve into various image similarity techniques and demonstrate how to implement them using Python.

Understanding Image Similarity

Image similarity can be thought of as a numerical representation of how alike two images are in terms of their visual content. There are several dimensions along which images can be similar, such as color, shape, texture, and composition. Various mathematical and computational methods are employed to quantify these similarities, enabling us to compare and categorize images efficiently.

Popular Image Similarity Approaches

Histogram-Based Approaches

Histograms capture the distribution of pixel values in an image. By comparing the histograms of two images, you can measure their similarity.

The Histogram Intersection and Histogram Correlation metrics are commonly used for this purpose. Python's ***OpenCV*** library provides tools to compute and compare histograms.

Here, we'll demonstrate a simple example using Histogram-Based Approaches and the ***opencv-python*** library:

We'll be using these two images in the rest of the blog's demonstration parts.

```
import cv2
# Load images
image1 = cv2.imread(image1)
image2 = cv2.imread(image2)
hist_img1 = cv2.calcHist([image1], [0, 1, 2], None, [256, 256, 256], [0, 256, 0,
hist_img1[255, 255, 255] = 0 #ignore all white pixels
cv2.normalize(hist_img1, hist_img1, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX)
hist_img2 = cv2.calcHist([image2], [0, 1, 2], None, [256, 256, 256], [0, 256, 0,
hist_img2[255, 255, 255] = 0  #ignore all white pixels
cv2.normalize(hist_img2, hist_img2, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX)
# Find the metric value
metric_val = cv2.compareHist(hist_img1, hist_img2, cv2.HISTCMP_CORREL)
print(f"Similarity Score: ", round(metric_val, 2))
# Similarity Score: 0.94
```

Structural Similarity Index (SSIM)

SSIM is a widely used metric that assesses the structural similarity between two images. It considers luminance, contrast, and structure, giving a score between -1 (dissimilar) and 1 (identical). The scikit-image library in Python offers an SSIM implementation.

Here, we'll demonstrate a simple example using SSIM and the ***scikit-image*** library:

```
import cv2
from skimage import metrics
# Load images
image1 = cv2.imread(image1)
image2 = cv2.imread(image2)
image2 = cv2.resize(image2, (image1.shape[1], image1.shape[0]), interpolation =
print(image1.shape, image2.shape)
# Convert images to grayscale
image1_gray = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)
image2_gray = cv2.cvtColor(image2, cv2.COLOR_BGR2GRAY)
# Calculate SSIM
ssim_score = metrics.structural_similarity(image1_gray, image2_gray, full=True)
print(f"SSIM Score: ", round(ssim_score[0], 2))
# SSIM Score: 0.38
```

The main drawback of the SSIM approach compared to the Histogram approach is that the images have to be the same dimension. Even the similarity score is very low. We can do background subtraction and transparency removal from the images to improve the similarity score.

Feature-Based Approaches

These methods extract salient features from images, such as edges, corners, or key points. Techniques like Scale-Invariant Feature Transform (SIFT) and Speeded-Up Robust Features (SURF) identify distinctive points in images, which can then be compared across images.

The ***opencv-python*** library can be used for SIFT and SURF.

Deep Learning-Based Approaches

Deep learning has revolutionized image similarity tasks. Using pre-trained convolutional neural networks (CNNs) like ResNet, VGG, and Inception, you can extract deep features from images.

CLIP (Contrastive Language-Image Pre-Training) from the openAI is an impressive multimodal zero-shot image classifier that achieves impressive results in a wide range of domains with no fine-tuning. It applies the recent advancements in large-scale transformers like GPT-3 to the vision arena.

We can fine-tune these models on our own image and text data with the regular SentenceTransformers training code. **ScrapeHero** helps in preparing your own image dataset to train these models. Its web crawling service can crawl complex websites and provides high-quality data.

Here, we'll demonstrate a simple example using CLIP-Based pre-trained model and the ***torch, open_clip*** & ***sentence_transformers*** libraries:

```
!pip install git+https://github.com/openai/CLIP.git
!pip install open_clip_torch
!pip install sentence_transformers

import torch
import open_clip
import cv2
from sentence_transformers import util
from PIL import Image
# image processing model
device = "cuda" if torch.cuda.is_available() else "cpu"
model, _, preprocess = open_clip.create_model_and_transforms('ViT-B-16-plus-240'
model.to(device)
def imageEncoder(img):
    img1 = Image.fromarray(img).convert('RGB')
    img1 = preprocess(img1).unsqueeze(0).to(device)
    img1 = model.encode_image(img1)
    return img1
def generateScore(image1, image2):
    test_img = cv2.imread(image1, cv2.IMREAD_UNCHANGED)
    data_img = cv2.imread(image2, cv2.IMREAD_UNCHANGED)
    img1 = imageEncoder(test_img)
    img2 = imageEncoder(data_img)
    cos_scores = util.pytorch_cos_sim(img1, img2)
    score = round(float(cos_scores[0][0])*100, 2)
    return score
print(f"similarity Score: ", round(generateScore(image1, image2), 2))
#similarity Score: 76.77
```

The similarity between images can then be computed based on the cosine similarity or Euclidean distance of these feature vectors. To improve the accuracy, we can preprocess the images

Applications

The main applications of the image similarity technique include e-commerce product matching, image retrieval, object recognition, and face recognition. Image similarity, for example, is used in image retrieval to find images similar to a query image.

Image similarity can be used in object recognition to match a given object with a known database. An image similarity algorithm is used to identify people by comparing their faces to a database.

Conclusion

The ability to measure image similarity is a vital component of numerous applications in today's visually driven world. This blog has introduced you to various image similarity approaches, from simple histogram-based methods to complex deep-learning techniques.

You can also explore the **Siamese networks**, a special class of neural networks designed for one-shot learning and image similarity tasks.

Python, with its rich ecosystem of libraries like ***scikit-image***, ***opencv-python***, ***TensorFlow***, and ***PyTorch***, empowers developers and researchers to implement these approaches effectively.

Experimenting with these techniques will open doors to creating innovative applications that leverage the power of image similarity.

If you've found this article helpful or intriguing, don't hesitate to give it a clap! As a writer, your feedback helps me understand what resonates with my readers.

Follow ScrapeHero for more insightful content like this. Whether you're a developer, an entrepreneur, or someone interested in web scraping, machine learning, AI, etc., ScrapeHero has compelling articles that will fascinate you.

Image Processing · Image Similarity · Python · Data Science · Hugging Face

Written by Vasista Reddy

Follow

172 Followers · Writer for ScrapeHero

Works at Cognizant. Ex-Turbolab-ian and loves trekking.... Reach_Me_Out_on_Linkedin: https://www.linkedin.com/in/vasista-reddy-100a852b/