Image Similarity Search using Feature Detection and Spotify Annoy

Introduction

In this project, we build an image similarity search tool using deep learning and approximate nearest neighbors. We leverage a pre-trained ResNet18 model to extract image features and Spotify's Annoy library to perform efficient similarity search. This method allows us to find images that are visually similar to a given query image.

Understanding Feature Detection

Feature detection refers to identifying and extracting important visual patterns from an image that help differentiate one image from another. These patterns are often high-level representations (edges, textures, shapes, etc.) captured by deep neural networks. In our case, the ResNet18 model, trained on the ImageNet dataset, is used to extract a 512-dimensional feature vector that uniquely describes the content of an image.

By removing the final classification layer of ResNet18, we obtain pure feature vectors that describe images without assigning a class. These features are then used for similarity comparison.

How the System Works

1. Load a pre-trained ResNet18 model and strip off its final classification layer.
2. Preprocess all dataset images (resize, normalize, etc.).
3. Pass each image through the model to get a 512-dimensional feature vector.
4. Add all feature vectors to an Annoy index.
5. For a given query image, extract its features and retrieve the most similar images from the index using angular distance.

Python Code

```
import os
import torch
from torchvision import models, transforms
from PIL import Image
from annoy import AnnoyIndex

model = models.resnet18(pretrained=True)
model.fc = torch.nn.Identity()
model.eval()

transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(
        mean=[0.485, 0.456, 0.406],
        std=[0.229, 0.224, 0.225]
    )
])

def extract_feature(img_path):
    image = Image.open(img_path).convert("RGB")
    image = transform(image).unsqueeze(0)
    with torch.no_grad():
        features = model(image)
    return features.squeeze().numpy()

feature_dim = 512
ann_index = AnnoyIndex(feature_dim, 'angular')

image_paths = [f"dataset/{img}" for img in os.listdir("dataset") if img.endswit

for idx, img_path in enumerate(image_paths):
    vec = extract_feature(img_path)
    ann_index.add_item(idx, vec)

ann_index.build(10)
ann_index.save("image_features.ann")

query_vec = extract_feature("query.jpg")
nearest_ids = ann_index.get_nns_by_vector(query_vec, 5)

for i in nearest_ids:
    print(image_paths[i])
```

Conclusion

This project showcases the power of pre-trained models and efficient indexing algorithms for real-world applications. With just a few lines of code, we can build a scalable and fast image search engine using PyTorch and Spotify Annoy.