2248-CSE 6363-002 - MACHINE LEARNING – HW3 REPORT

Team Members

Nikhil Reddy Duggireddy-1001968033
Laya Kalali-1002128876
Sasikala Paturu-1002083123
Lokeshwar Kodipunjula-1002175121

IMAGE RETRIEVAL USING CIFAR-10 DATASET:

INTRODUCTION:

Image retrieval is a technique used to search and retrieve images from a large dataset based on visual content. It is commonly applied in various fields, including e-commerce, social media, and digital archives, where users want to find similar images to a query image. This project demonstrates an image retrieval system using the CIFAR-10 dataset, a widely used dataset in image classification and retrieval tasks. By leveraging deep learning and feature extraction with a pre-trained convolutional neural network (CNN), we aim to retrieve images that are visually like a given query image.

DATASET OVERVIEW:

The CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test-images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.
For this project, we resize the images to fit the input requirements of a pre-trained ResNet50 model, which requires a minimum resolution of 32x32.

Here are the classes in the dataset, as well as 10 random images from each:
Airplane, Automobile (cars and trucks), **Bird, Cat, Deer, Dog, Frog, Horse, Ship, Truck.**

These classes represent common objects, making CIFAR-10 a relevant dataset for general object recognition tasks.

METHODOLOGIES:

1.Preprocessing: We resize and preprocess the images to prepare them for input to the ResNet50 model.
2. **Feature Extraction**: Using a pre-trained ResNet50 model, we extract high-dimensional feature vectors from the images, representing each image's unique characteristics.
3. **Image Retrieval**: We use a Nearest Neighbors algorithm with cosine similarity to find images like a given query image based on the extracted feature vectors.
4. **Display Results**: The retrieved images are displayed alongside the query image for easy comparison.

CODE SNIPPETS:

Below are key code snippets used in this project:

STEP 1: IMPORT LIBRARIES

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import NearestNeighbors
import tensorflow as tf
```

STEP 2: LOAD AND PREPROCESS DATASET

```
[ ] # Load CIFAR-10 dataset
    (x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar10.load_data()

    # Resize to target shape for ResNet50 input (64x64 images -> for lower memory usage)
    # Use method='bilinear' and antialias=True for better quality resizing
    x_train_resized = tf.image.resize(x_train, [32, 32], method='bilinear', antialias=True)
    x_test_resized = tf.image.resize(x_test, [32, 32], method='bilinear', antialias=True)


    # Preprocess images for ResNet50
    x_train_resized = tf.keras.applications.resnet.preprocess_input(x_train_resized)
    x_test_resized = tf.keras.applications.resnet.preprocess_input(x_test_resized)
```

STEP 3: EXTRACT FEATURES USING RESNET50

```
[ ] # Load ResNet50 model without the top layer for feature extraction
    base_model = tf.keras.applications.ResNet50(weights='imagenet', include_top=False, input_shape=(32, 32, 3))
    model = tf.keras.Model(inputs=base_model.input, outputs=base_model.output)

    # Extract features for training images
    train_features = model.predict(x_train_resized, batch_size=64)
    train_features = train_features.reshape(train_features.shape[0], -1)  # Flatten to 1D

    # Extract features for test images
    test_features = model.predict(x_test_resized, batch_size=6)
    test_features = test_features.reshape(test_features.shape[0], -1)  # Flatten to 1D
```

STEP 4: IMPLEMENT NEAREST NEIGHBORS FOR IMAGE RETRIEVAL

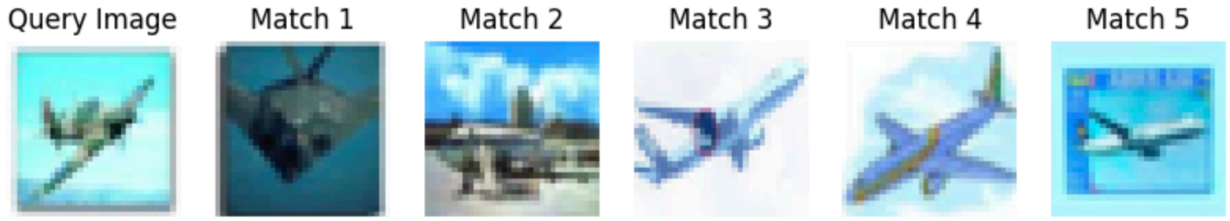
```
[ ]  # Fit Nearest Neighbors model
     neighbors = NearestNeighbors(n_neighbors=5, metric='cosine')
     neighbors.fit(train_features)

     # Define a function to retrieve similar images
     def retrieve_similar_images(query_index):
         query_feature = test_features[query_index].reshape(1, -1)
         distances, indices = neighbors.kneighbors(query_feature)

         # Plot the query image
         plt.figure(figsize=(10, 3))
         plt.subplot(1, 6, 1)
         plt.imshow(x_test[query_index])
         plt.title("Query Image")
         plt.axis('off')

         # Plot the retrieved images
         for i, idx in enumerate(indices[0]):
             plt.subplot(1, 6, i + 2)
             plt.imshow(x_train[idx])
             plt.title(f"Match {i + 1}")
             plt.axis('off')
         plt.show()
```

STEP 5: TEST THE IMAGE RETRIEVAL SYSTEM

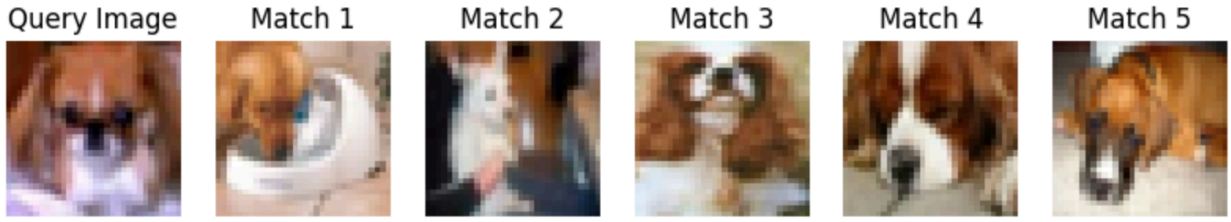
```
# Test retrieval for a random test image
query_index = np.random.randint(0, len(x_test))
retrieve_similar_images(query_index)
```

EXPERIMENTS AND RESULTS:

Query Image: Image of a frog
- **Retrieved Images**: Images of frog with similar poses and colors



Query Image: Image of an airplane
- **Retrieved Images**: Images of airplanes, some with similar color schemes or angles



DISCUSSION:

The image retrieval system effectively retrieved images that were visually like the query images, demonstrating the potential of deep learning-based feature extraction for image similarity tasks. However, limitations include:

1. **Low Resolution**: CIFAR-10 images are low-resolution, making it challenging to capture fine-grained details.
2. **Computational Complexity**: Extracting high-dimensional feature vectors and performing nearest-neighbor search on them can be computationally intensive. More efficient techniques, such as approximate nearest neighbors, may improve the system's scalability.
3. **Feature Representation**: Using a pre-trained model on a different dataset (ImageNet) may limit performance, as the model is not fine-tuned for CIFAR-10 classes.

Future work could involve experimenting with different pre-trained models or fine-tuning a model specifically for CIFAR-10 to enhance feature quality.

REFERENCE:

- **Dataset**: Krizhevsky, A., & Hinton, G. (2009). *_Learning multiple layers of features from tiny images_*. CIFAR-10 dataset.
- **Pre-trained Model**: He, K., Zhang, X., Ren, S., & Sun, J. (2016). *_Deep Residual Learning for Image Recognition_*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
- **Nearest Neighbors**: Pedregosa, F., et al. (2011). *_Scikit-learn: Machine Learning in Python_*. Journal of Machine Learning Research, 12, 2825–2830.