Feature Extraction and PCA-based Encoding for Pet Images

Rithvik Koruturu

Apr 2025

1 Introduction

This document explains a Python script for extracting features from grayscale pet images using the Scale-Invariant Feature Transform (SIFT) and encoding them using Principal Component Analysis (PCA). The dataset consists of images of cats and dogs, which are processed, resized, and converted into a reduced feature space.

2 Importing Required Libraries

```
import cv2
import numpy as np
import os
import pickle
from tqdm import tqdm
from sklearn.decomposition import PCA
```

- **OpenCV (cv2)**: Used for image loading, resizing, and SIFT feature extraction. - **NumPy**: Used for numerical computations and array manipulation. - **OS**: Handles file path operations. - **Pickle**: Saves extracted feature data. - **tqdm**: Provides a progress bar for file processing. - **scikit-learn PCA**: Used for dimensionality reduction.

3 Dataset Configuration

```
dataset_path = "D:/datasets/PetImages"
categories = ["Cat", "Dog"]
image_size = (255, 255)
num_images_per_category = 1000
```

- The dataset is stored in `"D:/datasets/PetImages"`. - The script processes **1000 images per category**.

4 Feature Extraction using SIFT

4.1 SIFT Feature Extraction Function

```
def extract_features(image):
    sift = cv2.SIFT_create()
    keypoints, descriptors = sift.detectAndCompute(image, None)
    if descriptors is None:
        return np.zeros((1, 128))  # Default zero vector
    return descriptors.mean(axis=0)  # Mean feature vector
```

- SIFT extracts keypoints and descriptors from the image. - If no descriptors are found, it returns a zero vector. - Otherwise, the function returns the **mean feature vector** of all detected descriptors.

5 Processing Images

```
encoded_data = {}

for category in categories:
    path = os.path.join(dataset_path, category)
    encoded_data[category] = []
    image_files = [f for f in os.listdir(path) if f.endswith(".jpg")][:num_imag

    print(f"Encoding {category} images...")
    for image_file in tqdm(image_files):
        try:
            image_path = os.path.join(path, image_file)
            image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

            if image is None:
                print(f"Skipping {image_file}, couldn't read the image.")
                continue

            image_resized = cv2.resize(image, image_size)
            features = extract_features(image_resized)
            encoded_data[category].append(features)

        except Exception as e:
            print(f"Error processing {image_file}: {e}")
```

- Each image is: - Loaded in grayscale. - Resized to 255×255 pixels. - Processed with SIFT for feature extraction. - Stored in a dictionary under its respective category.

6 Converting Data to NumPy Arrays

```
for category in categories:
    encoded_data[category] = np.array(encoded_data[category])
```

- Ensures that extracted features are converted into `NumPy` arrays.

7 Applying PCA for Dimensionality Reduction

```
for category in categories:
    num_samples, num_features = encoded_data[category].shape
    n_components = min(100, num_samples, num_features)  # Max 100 or lower if ne

    if num_samples > 0:
        pca = PCA(n_components=n_components)
        encoded_data[category] = pca.fit_transform(encoded_data[category])
```

- PCA is applied dynamically: - The number of components is set to min(100, num_samples, num_features).
- Ensures that PCA does not exceed the number of available samples or features.

8 Saving the Encoded Data

```
with open("quantum_encoded_data.pkl", "wb") as f:
    pickle.dump(encoded_data, f)

print("Successfully encoded with OpenCV & PCA!")
```

- Saves the encoded feature vectors into a **pickle** file.

9 Conclusion

This script efficiently extracts features from pet images using SIFT and applies PCA to reduce the dimensionality of feature vectors. The final encoded dataset can be used for further machine learning tasks such as classification.