CSE515 - Multimedia and Web Databases - Fall 2023

Project Report - Phase 1

Kaushik Ravishankar
Madhura Bhalchandra Wani
Mohankrishna Chandrashekar
Niraj Sonje
Pavan Rathnakar Shetty
Pranav Borikar

Abstract

In the realm of multimedia analysis, representation of data is crucial to understanding any problem at hand. This initial project phase aimed to acquaint our team with the fundamental tools, datasets and feature models that are commonly used in this field. Using Python, PyTorch, a framework tailor-made for deep learning, TorchVision, a specialized tool for computer vision tasks, essential mathematical libraries like scipy and numpy, we operate on the simple but comprehensive Caltech 101 dataset. We use the ResNet50 pre-trained neural architecture, extracting features from its avgpool, layer3, and fc layers. In particular, we calculate the color moments and histogram of oriented gradients (HOG) feature representations adjusted specifically for the task. The feature descriptors processed from the dataset are stored in a MongoDB database for unfettered access. These feature descriptors are essential in identifying similar images using various similarity measures, such as Euclidean distance, cosine similarity, and Pearson correlation coefficient.

Keywords

Image processing, feature extraction, feature descriptors, color moments, histogram of oriented gradients, visualization, ResNet50, Caltech101, MongoDB, Euclidean distance, cosine similarity, Pearson similarity.

Introduction

Key Terminology

- *Features* - Unique attributes or traits derived from data, frequently employed for tasks such as analysis, identification, and categorization
- *Feature descriptors* - Representations of features in numerical, vectorized formats
- *Caltech101* - A dataset of 101 diverse categories of images, which was popularly used in object recognition benchmarks

- *Color moments* - Statistical properties of color distributions within an image, such as mean, standard deviation and skewness, providing insights into its color composition and distribution
- *Histogram of oriented gradients (HOG)* - Weighted distribution of texture gradient orientations in an image, providing insights into texture distribution and orientations
- *ResNet50* - A residual neural network architecture which has 50 layers, exceptionally performant in image classification tasks, trained on the ImageNet 1K dataset
- *Distance/similarity measure* - Quantifier of the dissimilarity or similarity between two data points, helping to assess how close or different they are in a given feature space, commonly used for clustering, classification, and retrieval tasks in multimedia analysis. Examples include

Goal Description

In this initial phase of the project, the overall objective is to identify and visualize the most similar images of a given image from the Caltech101 dataset based on five different feature models, namely: color moments, histogram of oriented gradients, and three kinds of features extracted from the avgpool, layer3 and fc layers of the pre-trained ResNet50 neural network. To achieve this, several similarity measures are experimented with and appropriately selected based on their performance on each feature model, such as Euclidean distance, cosine similarity, Pearson correlation.

Assumptions

The bulk of the dataset consists of images that have three color channels, namely red, green and blue (RGB). However a small portion of the dataset consists of grayscale images. To avoid inaccuracies and inconsistencies in comparisons, we will skip the processing of and feature extraction from these images wherever appropriate.

Proposed solution/Implementation

Environment, dataset and tools:

The environment uses Jupyter Notebooks on a PC preferably on a CUDA-enabled GPU for faster processing. The Caltech101 dataset and ResNet50 model can both be downloaded using modules from TorchVision.

Task 1: Feature descriptor modeling:

Accessing the dataset is done using TorchVision's Caltech101 class object, making the dataset subscriptable.

1. Color moments:
   - The given image is resized to the dimensions of 300x100, and then divided into 10x10 grids, with each grid having dimensions 30x10.
   - The numpy library is used to compute the mean and standard deviation, and the scipy library is used to calculate the skewness of each of the 100 smaller grids.

- These are then combined to form a feature descriptor of dimensional shape (10,10,3,3) i.e. 900 dimensions
- These color moments are visualized as a stacked bar chart representing the color moments of each channel together for each 30 x 10 grid. (Note: nan values of skewness do not have bars)

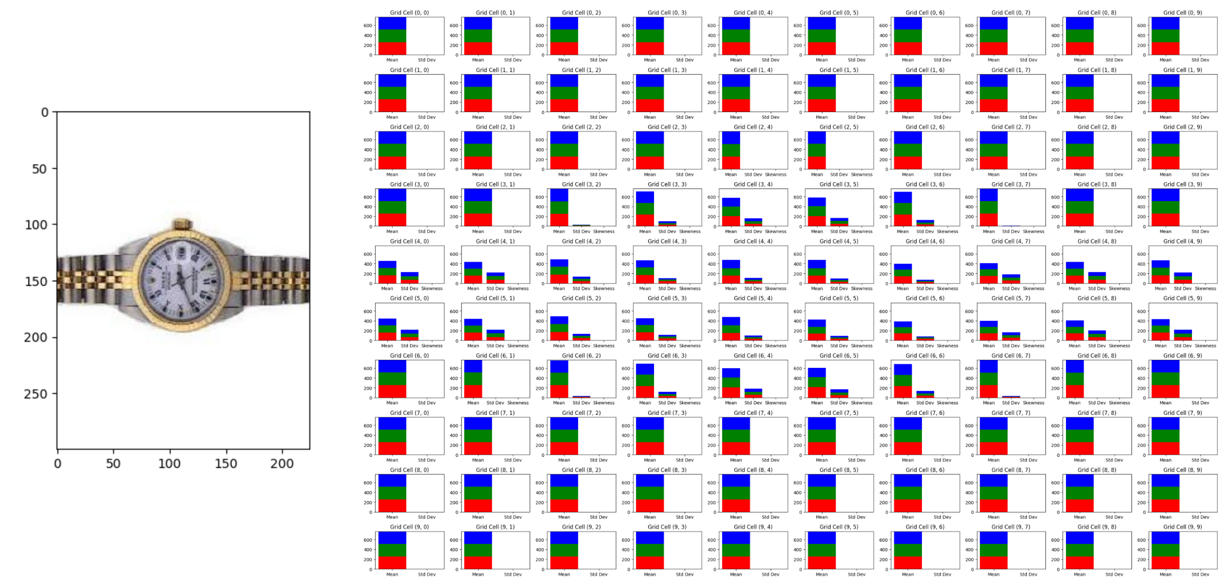
Fig 1: Example visualization of color moments as a grid of stacked bar charts. Visual resemblance can be observed

2. Histogram of oriented gradients (HOG):
    - The given image is resized to the dimensions of 300x100, and then divided into 10x10 grids, with each grid having dimensions 30x10.
    - The entire image is converted to grayscale using the OpenCV library, to facilitate gradient calculation.
    - The gradient change in horizontal and vertical directions are calculated using the Sobel operator of the OpenCV library, using the first-order central difference option of the function, using the kernel masks $dx = [-1, 0, 1]$ and $dy = [-1, 0, 1]^T$ for convolving with the image grid to get the gradients.
    - The magnitude and direction in degrees of the grid's gradient are computed, and the gradients are binned by their direction into 9 bins of equal ranges comprising a total range of $(-180°, 180°)$ and weighted by their magnitude to form a histogram.
    - These are then combined to form a feature descriptor of dimensional shape (10,10,9) i.e. 900 dimensions

- This HOG is visualized as histograms themselves, as well as an overlay grid of arrows representing the dominant gradients of each cell

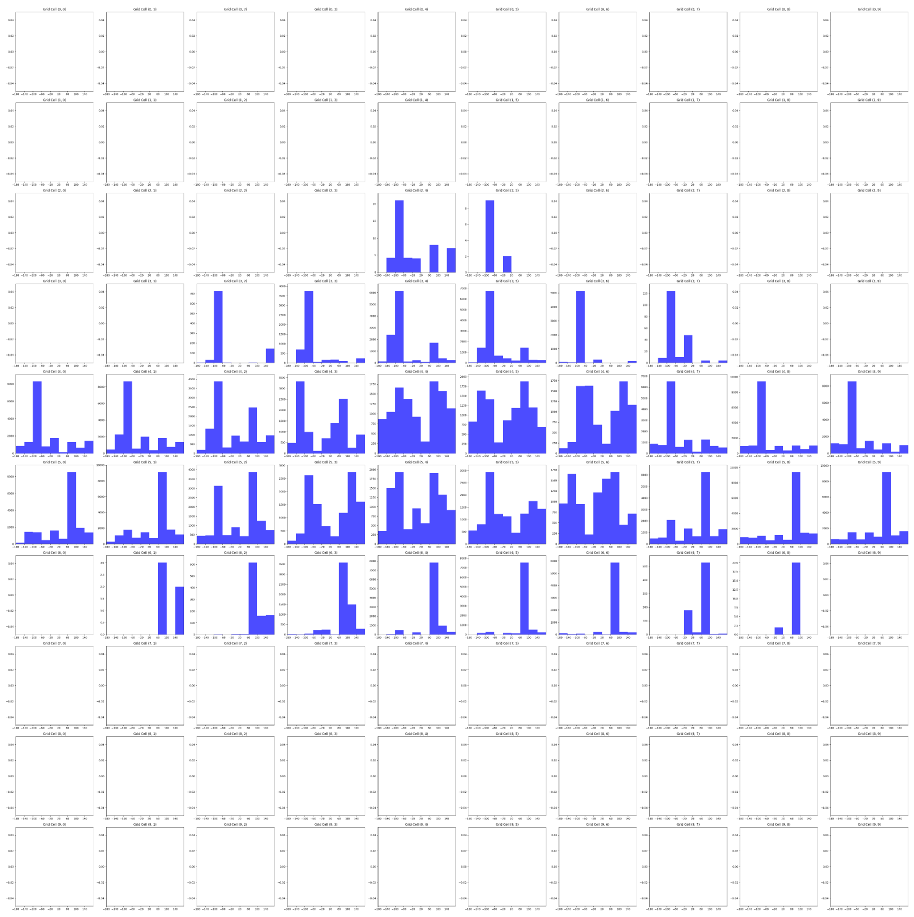

Figure 2: Example visualization of HOG as histograms themselves

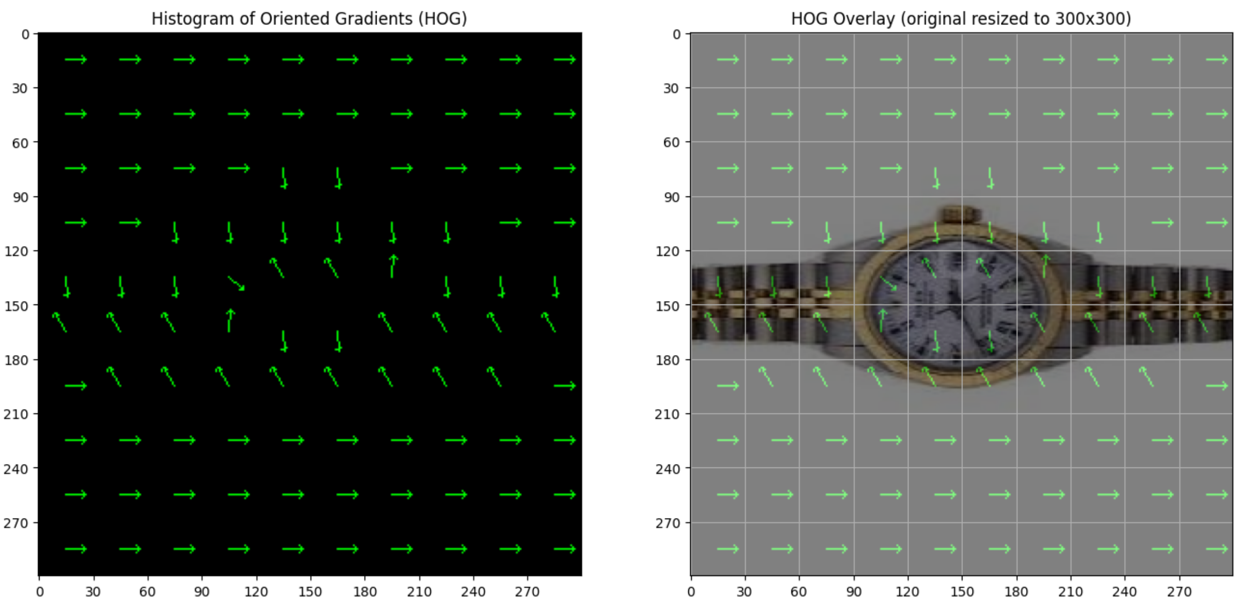

Figure 3: Example visualization of HOG as a grid of arrows representing the dominant gradient directions in each grid cell. Note that the all-white areas cause zero gradient and have zero orientation, hence the rightward arrows.

3. ResNet50 - avgpool, layer3 and fc layers
    - The pre-trained ResNet50 model is loaded with its default weights using the TorchVision module.
    - Both the model and image data are loaded into the GPU memory using torch functionality itself, for performance benefits.
    - The given image is resized to the dimensions of 224x224, the shape of the input layer of the neural network.
    - To the avgpool, layer3 and fc layers of the neural net, hooks that capture the output of the layer are attached.
    - The feature descriptor of size 2048 obtained from the avgpool layer is shrunk to size 1024 by averaging consecutive pairs of elements.
    - The feature descriptor of size 1024x14x14 obtained from the layer3 layer is shrunk to size 1024 by averaging along the 14x14 grids
    - The feature descriptor of size 1000 is obtained from the fc layer.

Task 2: Dataset feature extraction and storage:

Using the feature models developed in the previous task, features are extracted from the entire dataset and stored in a local instance of a MongoDB database. The MongoDB database engine allows for efficient and easy data storage and retrieval. In this case, NoSQL is preferred to relational databases due to their inefficiency in storing large binary blobs and complexity when handling inconsistent data structures.

Direct binary file storage is also not a good choice due to preferred isolation of data from code, integrity and scalability concerns. Moreover, MongoDB provides a very smooth developer experience, and allows easy conversion between in-memory and database structures. The PyMongo module provides the database driver functionality between Python and MongoDB.

Task 3: Similarity measures

The following similarity metric - feature model pairs were chosen for our project:

1. *Color moments* - Pearson Correlation: Pearson correlation accounts for covariance in features, and is not susceptible to outliers. Useful since color distribution is highly varied and "spotty"

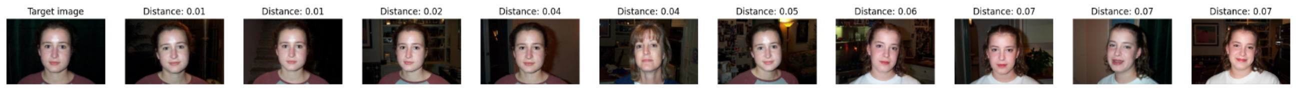

Figure 4: 10 similar images for image ID 123 using Pearson similarity measure

2. *HOG* - Cosine similarity: By definition cosine similarity is useful to compare angles and directional differences, and a HOG is built on gradient orientations.

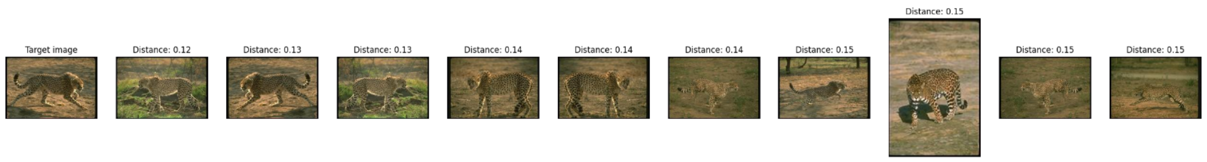

Figure 5: 10 similar images for image ID 880 using cosine similarity measure

3. *ResNet50 layers* - Euclidean distance

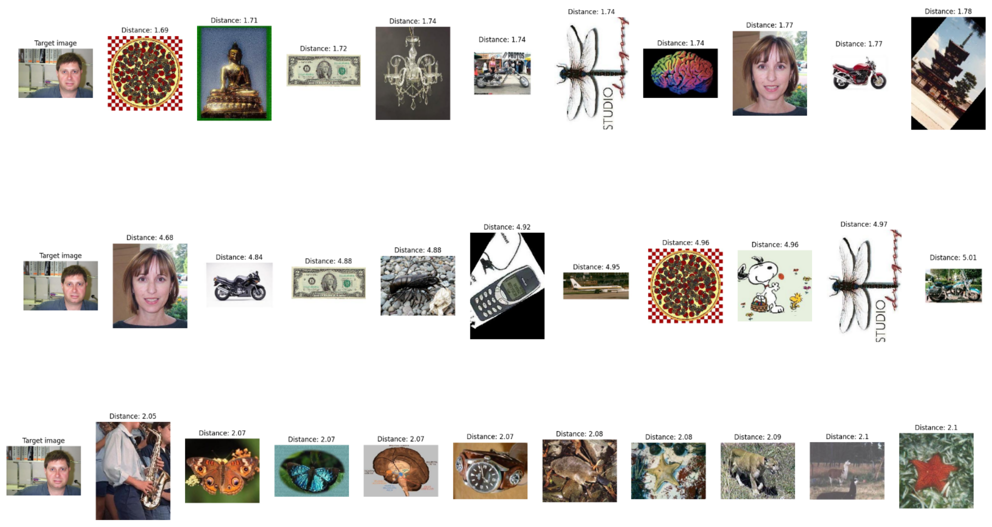

Figure 6: Euclidean distance measure used to find 10 similar images for image ID 0 for the avgpool, layer3 and fc feature models

Interface specification

The runtime is the Jupyter notebook environment, allowing for step-through execution, cell by cell. It is also easily interactive in displaying images and getting user inputs. Matplotlib library is used to visualize images.

System requirements:

- The code is developed keeping a GPU-available system in mind. Minor modifications would be needed in case of execution on a non-GPU system, for the ResNet50 feature extraction part.
- Necessary Python modules are listed in the requirements.txt file
- MongoDB installation is required, either local or hosted/Atlas. Connection string may have to be modified as needed if not local.
- Configuration of dataset path is needed as per host machine's directory system

Conclusion

In this phase of the project, we have successfully developed an experimental system for image similarity analysis using feature models such as color moments, histogram of gradients (HOG), and a pre-trained ResNet50 model, utilizing efficient storage and retrieval mechanisms. We observed that the feature extraction for image ID 0 produced results that differed from our expectations. This discrepancy highlights our uncertainty surrounding the ResNet feature models. However, color moments and HOG together provide satisfactory results for most of the images in the dataset.

Appendix

For the most part, we worked separately. Some of us were not as familiar with Python and image processing techniques. However we worked together to overcome our limitations and independently worked on the tasks. But there was active discussion regarding choice of similarity measures, database engine and other troubleshooting activities. We all played equal parts in achieving this goal.