REPORT

Objective: To find duplicate images in the dataset.

Approaches

1. Hashing: All Images are hashed and similar hashes are matched. This process is fast, but only works if the images are exactly matched and cannot detect flipped, rotated images, blur images.
2. Hamming Code (Fingerprinting): In this method I am using the Hamming code difference with a threshold to file the differences, this process is very time consuming as the difference for all the images have to calculated to find the similarity for 1 image. While this process is able to find images that are blurred it cannot find images that are rotated or flipped.
3. Using a pre-trained model: In this method I have used a pretrained model in this case "VGG16" to extract features from each image and use Nearest Neighbour to find the closest representation to the source image. This method is good as it is able to extract features and is able to find duplicates even if the images are flipped or rotated, also this provides a more "**Generalised**" solution to the problem.

The following assumptions or considerations have been made for the dataset provided,

1. All images belong to a particular category.
2. Only a portion of the dataset is using around 50000 Images.
3. All the techniques are unsupervised.
4. The Project is split into IPythonNotebooks and Python scripts.
5. I was not able to produce the submission files for "Feature Extraction" as my system could not handle the process.

Files

1. Preprocess.py: Used to download images (Tops) from the csv file.
2. Analysis.ipynb: Simple analysis of the CSV file and to better understand the categories and size of images, particularly was used to determine the width of the image as 256, as it was the median for all images.
3. Hashing.ipynb: Used the Hashing method with a local databased to store the hashed values.
4. Fingerprints.ipynb: Used both Hashing and Hamming distance to find duplicate images, was not able to process Hamming duplicates due to lack of processing power.
5. Pre_train_model_fe.py: Implements the function of feature extraction and predicting the nearest neighbour for the entire dataset.
6. Inference.py: Implements inference for new files to which duplicates are to be predicted.

With more compute power and storage, I would have used “Variational Autoencoders” as it would have proved much better to use the latent space of the model as features for each image, as pretrained model (VGG16) is trained to extract features from ImageNet, the same feature space might not be an ideal match for this dataset and inference usually means prediction of latent representations given new, never-before-seen datapoints.

This can be further improved by using a Hybrid of the VAE and Object-detection model to find duplicates/similar images in a large image.