Imagededup

CNN Method:

1. Initialization:

- Sets up the CNN model (MobileNetV3 by default) for feature extraction.
- Initializes the PyTorch model, device (GPU if available, else CPU), and other configurations.

2. Encoding Generation:

- `**encode_image**`: Generates CNN encoding for a single image or whole image directory

- `encoding = myencoder.encode_image(image_file='path/to/image.jpg')`
- OR
- `encoding = myencoder.encode_image(image_array=<numpy array of image>)`

3. Duplicate Detection:

- `**find_duplicates**`: Finds duplicates using pre-generated encodings or images in a directory.
- `duplicates = myencoder.find_duplicates(image_dir='path/to/directory', min_similarity_threshold=0.85, scores=True, outfile='results.json')`

- `**find_duplicates_to_remove**`: Returns a list of duplicate image file names based on a similarity threshold.

- `duplicates = myencoder.find_duplicates_to_remove(image_dir='path/to/images/directory'),min_similarity_threshold=0.85)`

- **'_find_duplicates_dict'**: A dictionary where each key is a filename and the corresponding value is a list of duplicate filenames. If the scores parameter is set to True, the value will be a list of tuples, where each tuple contains the duplicate filename and the cosine similarity score. If scores is False, the list will contain only the duplicate filenames.

- `if scores is True, then a dictionary of the form {'image1.jpg': [('image1_duplicate1.jpg',score), ('image1_duplicate2.jpg', score)], 'image2.jpg': [] ..}`
- `if scores is False, then a dictionary of the form {'image1.jpg': ['image1_duplicate1.jpg','image1_duplicate2.jpg'],'image2.jpg':['image1_duplicate1.jpg',..], ..}`

4. Internal Methods:

- `**_validate_model_config**`: Validates the model configuration.
- `**_get_cnn_features_single**` and `**_get_cnn_features_batch**`: Methods to get CNN features for single or batched images.
- `**_check_threshold_bounds**`: Checks if the similarity threshold is within valid bounds.

Hashing Method:

- `**Hashing**`: Base class for hashing algorithms. It provides common functionality for encoding images and finding duplicates.

- `**PHash**`, `**AHash**`, `**DHash**`, `**WHash**` Classes: Subclasses that implement specific hashing algorithms (perceptual hashing, average hashing, difference hashing, and wavelet hashing).

- `**hamming_distance**` Method: Calculates the Hamming distance between two hash strings.

- `**_array_to_hash**` Method: Converts a matrix of binary numerals to a 64-character hexadecimal hash string.

- `**encode_image**` Method: Generates a hash for a single image.

- `**encode_images**` Method: Generates hashes for all images in a directory.

- `**find_duplicates**` Method: Finds duplicates either using the encoding mapping generated previously or using a directory of images.

- `**find_duplicates_to_remove**` Method: Generates a list of image file names to remove based on a given hamming distance threshold.

Script:

```
# https://idealo.github.io/imagededup/


from imagededup.methods import CNN
from imagededup.utils import plot_duplicates
import pandas as pd
import matplotlib.pyplot as plt
import os
```

```python
image_dir = "Images/"

from imagededup.methods import PHash
phasher = PHash()

# Generate encodings for all images in an image directory
encodings = phasher.encode_images(image_dir=image_dir)

# Find duplicates using the generated encodings
duplicates = phasher.find_duplicates(encoding_map=encodings)



cnn = CNN()



from imagededup.methods import CNN

cnn_encoder = CNN()

# Find duplicates using the generated encodings
duplicates_list = cnn_encoder.find_duplicates(image_dir=image_dir,
min_similarity_threshold=0.95, scores=True,
        outfile='results.json')

print(duplicates_list)

# plot_duplicates(image_dir=image_dir,
#                 duplicate_map=duplicates_list,
#                 filename='test.jpg')

for duplicate_filename in duplicates_list_cnn:
    # Create the full file path
    file_path = os.path.join(image_dir, duplicate_filename)
    # Delete the file
    os.remove(file_path)
    print(f"Deleted: {file_path}")
```

REFERENCE

https://idealo.github.io/imagededup/