[Samsung PRISM] Monthly Discussion 1

Filter Duplicate Images

Team

1. College Professors:
    2. Dr. Gayathri M / gayathrm2@srmist.edu.in
    3. Dr. M. Suganiya / suganiym@srmist.edu.in
4. Students:
    1. Kanupriya Johari / kg3878@srmist.edu.in
    2. Diptayan Jash / dj2037@srmist.edu.in
    3. Tuhina Tripathi / tt4102@srmist.edu.in
    4. Avya Rathod / ad0713@srmist.edu.in
5. Department: Department of Computing Technologies, SRM Institute of Science and Technology

Date: 28 Feb 2024

Filter Duplicate Images | Description

Problem Statement

Context

What might seem trivial and easy, Finding and weeding out duplicate images from a really large dataset is complex. It has to be done with high accuracy which can be difficult for multiple images.

Also, the method dictates memory consumption. To compare, if 1 image is compared with (n-1), the number of images within dataset and size of each image (as in hash method).

The worklet intends to investigate into various methods for quick and accurate duplication.

Statement

Filter Duplicate Images to optimize accuracy and memory consumption

Work let Details

6	4	Abhishek Mishra Ankit Mishra Athira Menon
Duration (Months)	**Members Count**	Mentors

Expectations

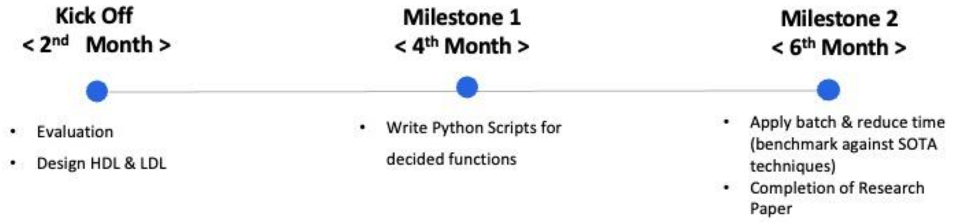
Undertaken Tasks

- Evaluate various image duplication checking and filtering methods including Hash, etc
- Evaluate Open-Source Scripts available & classify on basis of effectivity.
- Write custom script to find and filter out duplication in images.
- Test it for high scale and maintain accuracy.
- Improve the algorithm to improve the decided parameters.

KPI

- Write Research Paper stating innovative methods to find and filter duplication.
- Scalable, Production Ready Script
- Accuracy >98% on any given sample.

Timeline



Kick Off
< 2nd Month >

- Evaluation
- Design HDL & LDL

Milestone 1
< 4th Month >

- Write Python Scripts for decided functions

Milestone 2
< 6th Month >

- Apply batch & reduce time (benchmark against SOTA techniques)
- Completion of Research Paper

Last month's progress

- Conducted a thorough literature review of existing image duplication detection methods.
- Regularly discussed findings and progress in team meetings held through Google Meets.
- Recognized the complexity of balancing accuracy, efficiency and memory consumption in image duplication detection.
- Finalised hashing techniques.

Work-let Name: **Filter Duplicate Images**

Worklet Details

1. Worklet ID: 23RSG40SRM
2. College Name: SRM institute of Science and technology

KPIs achieved till now

1. Explored various hashing techniques, P-hash and D-hash, giving most effective results.
2. Implementing open-source scripts like OpenAI CLIP model for detection of duplicate images

Any Challenges/ Issues faced

1. Procuring a good enough dataset for testing and training models.
2. Removing the false duplicates and storing only the unique images.

Next Steps

1. Write custom scripts for filtering out duplicates using various methods.
2. Test scalability.

Key Achievements/ Outcome till now

1. Achieved filtering of duplicate images without heavy computation. Average time taken was 1 to 2 minute.

Date: 28 Feb, 2024

Image samples from the dataset

Dataset: California ND (704 total images)

Results and Observations

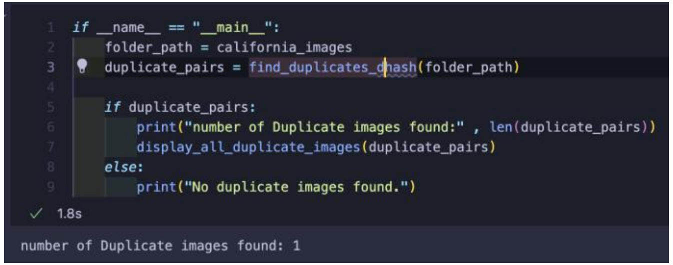
Work done	Results	Bottlenecks
1. Difference hashing (D-hash) 2. Perceptual hashing (P-hash) 3. OpenAI Clip Model	1. Works only on 1:1 (exact) duplicates. 2. Works on near duplicate images. 3. Takes a bit of time to find duplicates. 4. Works perfectly, immune to changes in brightness and contrast.	1. Sometimes the duplicates images are repeated and classified as false duplicates. 2. Clip model takes fairly large amount of time to compute. 3. P-hash and D-hash are affected by the changes in brightness and contrast.

Comparisons

D-hash	P-hash	Clip
● Works only on 1:1 duplicates i.e if the images are exact copy of each other. ● Very quickly calculates the duplicates.	● Works on near duplicates. If the images have same brightness and contrast. It successfully identifies the duplicates ● Average time: 20s.	● Works in all the cases. ● Average time: 4 min ● Depends upon the GPU we will be using.

RESULTS

D-hash



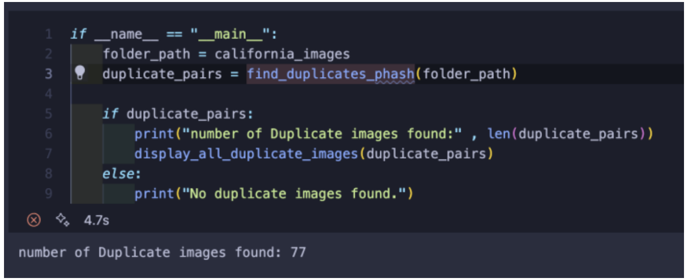
```
if __name__ == "__main__":
    folder_path = california_images
    duplicate_pairs = find_duplicates_dhash(folder_path)

    if duplicate_pairs:
        print("number of Duplicate images found:" , len(duplicate_pairs))
        display_all_duplicate_images(duplicate_pairs)
    else:
        print("No duplicate images found.")
```

✓ 1.8s

```
number of Duplicate images found: 1
```

P-hash



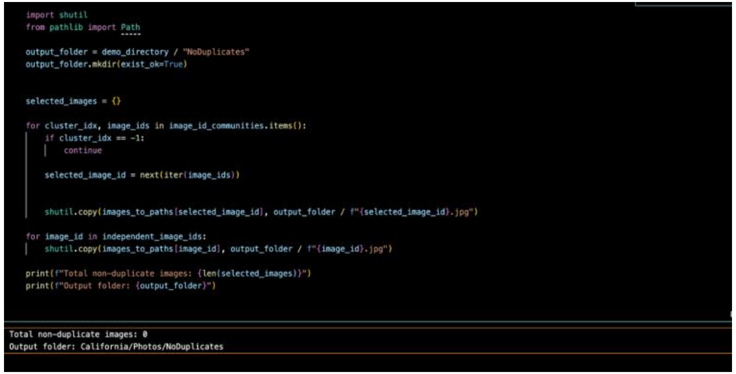
```
if __name__ == "__main__":
    folder_path = california_images
    duplicate_pairs = find_duplicates_phash(folder_path)

    if duplicate_pairs:
        print("number of Duplicate images found:" , len(duplicate_pairs))
        display_all_duplicate_images(duplicate_pairs)
    else:
        print("No duplicate images found.")
```

4.7s

```
number of Duplicate images found: 77
```

Clip - Stores images in the Non - Duplicates folder which now has 0 duplicates



```
import shutil
from pathlib import Path

output_folder = demo_directory / "NoDuplicates"
output_folder.mkdir(exist_ok=True)


selected_images = {}

for cluster_idx, image_ids in image_id_communities.items():
    if cluster_idx == -1:
        continue

    selected_image_id = next(iter(image_ids))


    shutil.copy(images_to_paths[selected_image_id], output_folder / f"{selected_image_id}.jpg")

for image_id in independent_image_ids:
    shutil.copy(images_to_paths[image_id], output_folder / f"{image_id}.jpg")

print(f"Total non-duplicate images: {len(selected_images)}")
print(f"Output folder: {output_folder}")
```

```
Total non-duplicate images: 0
Output folder: California/Photos/NoDuplicates
```