Fast and efficient image retrieval using binary hashes and network of word associations

Team Members :

Swarup Padhi (20CS30062)
Rounak Saha (20CS30043)
Saptarshi De Chaudhury (20CS10080)
Ritwik Ranjan Mallik (20CS10049)

Group: 15

Problem Statement

Task: Fast and efficient content based image retrieval using semantic similarity of images on CIFAR-100 dataset

Aim: Explore 2 methods:

(i) Neural hash codes for dimensionality and search space reduction

(ii) Network of word associations

Motivation:

(i) Capture semantic similarity in CBIR

(ii) Improvement in query retrieval time

Related Work

[1] Barz, B., & Denzler, J. (2019). **Hierarchy-based Image Embeddings for Semantic Image Retrieval**. IEEE Winter Conference on Applications of Computer Vision (WACV)

[2] Lin, K., Yang, H. F., Hsiao, J. H., & Chen, C. S. (2015). **Deep Learning of Binary Hash Codes for Fast Image Retrieval**. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop

[3] Palumbo, E., Allasia, W. (2015). S**emantic Similarity Between Images: A Novel Approach Based on a Complex Network of Free Word Associations**.

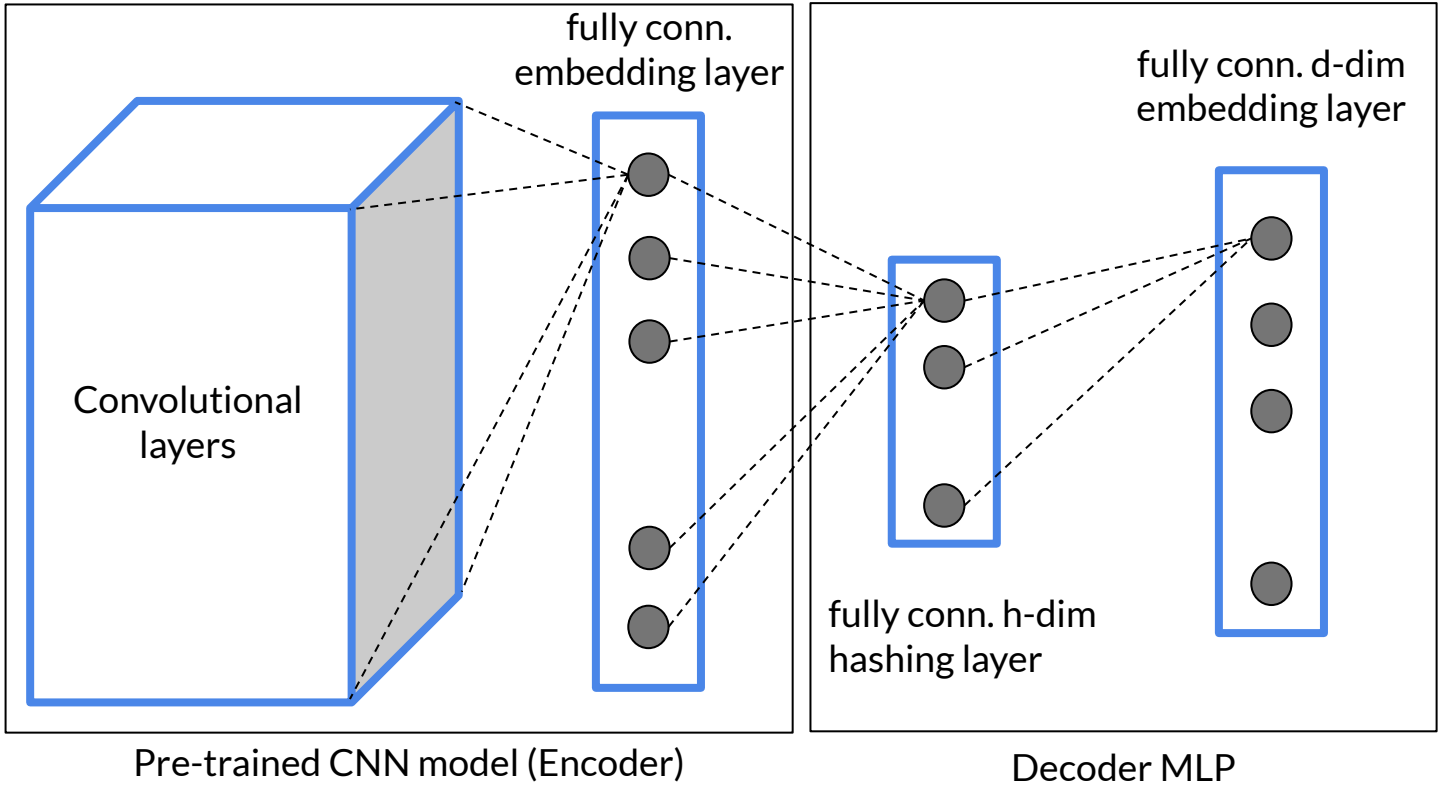
Class embeddings and semantic similarity

- We get 100-d class embeddings for the classes of CIFAR-100 from Barz et al. [1] Github link: https://github.com/cvjena/semantic-embeddings/
- Cosine similarity between embeddings of 2 classes reflects their **semantic similarity**
- We perform supervised training of a neural network model that predicts the **embedding** of the class an image belongs to when it is given as input

Binary hash codes: key idea

- Each image can be represented by h-bit binary hash code
- Hamming distance between two images reflects similarity
- Distance calculation is fast and efficient: simple bitwise XOR
- **We aim to learn binary hash codes of images from activations of a hidden layer of the neural network**

Model architecture

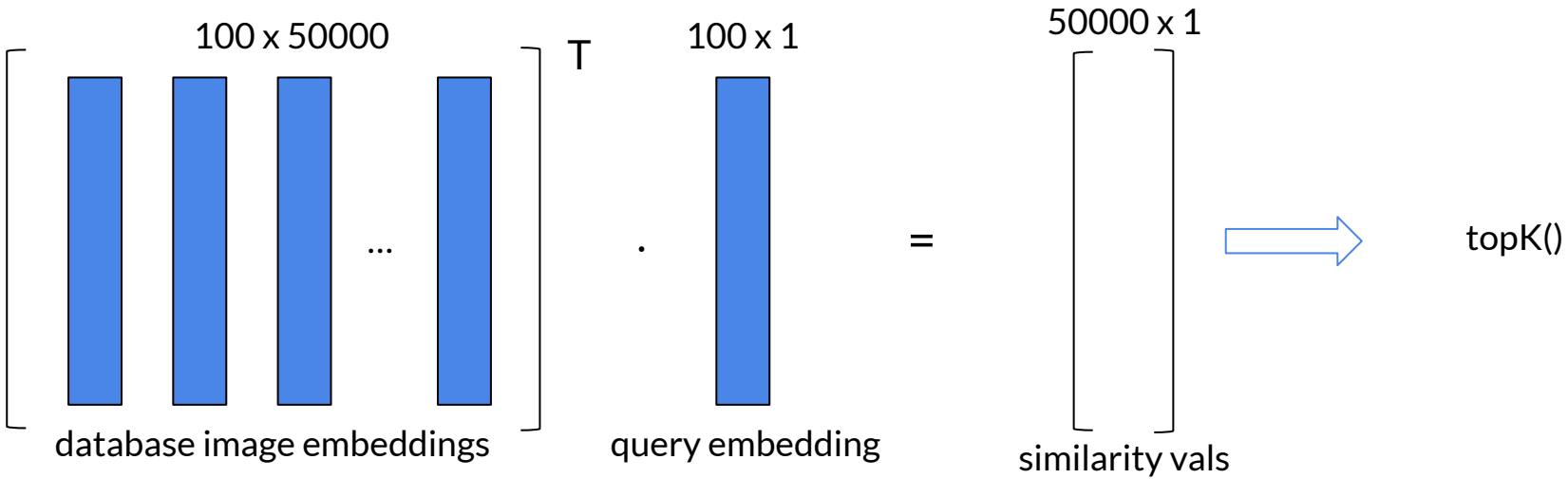


Model Hyperparameters

- Encoder
  - We use AlexNet CNN architecture in the encoder and add the decoder MLP instead of its final classification layer
  - The encoder output is 4096-d
- Decoder
  - The hash layer is 48-d (=*h*) with a sigmoid activation function
  - The final output layer is 100-d *(=d)* and predicts the embedding of class an image belongs to
- Training
  - Dataset: `<Input: image(224,224,3), output: class_embed(100,)>` for each record in the CIFAR-100 train set
  - We finetune the model with batch size 8 over 32 epochs using SGD optimizer with learning rate 0.001 and momentum 0.9 under the supervision of the simple cosine loss function

Exhaustive search

- k nearest neighbour search in high dimensional space
- Can be done efficiently by leveraging cosine similarity
- Matrix multiplication and topK() highly optimized in standard libraries



Speedup using binary hashes

- The hash layer is followed by sigmoid activation and hence output is always in {0,1}
- We binarize the value based on a threshold and use them as binary hashes
- Shortlist a small pool of candidate images according to Hamming distance of binary hashes and perform exhaustive search over that pool
- Optimization:
  - Bitwise operations fast
  - Smaller matrix multiplication

Experiments

For hierarchy-based neural hash codes (Algorithm-1, Algorithm-2, Algorithm-2+Hashes), we use the CIFAR-100 test images as query and corpus, and evaluate the performance on following metrics:

- Average query retrieval time
- Mean average precision mAP@k (k=1,5,10,15,250)
- Mean average hierarchical precision mAHP@k (k=1,5,10,15,250)

Code and Retrieval Results

```
embed_100_pdt = torch.matmul(img_dataset_embed_100, embed_100)

_, top1k_indices = torch.topk(embed_100_pdt.flatten(), k=topk)
```

Time [N =50k, d = 100, k = 250] = 0.0046 sec

Computes cosine similarity [Θ(Nd)]

Extract ***top k*** relevant images

Constant factor larger

```
_, top2k_indices = torch.topk(-torch.cdist(img_dataset_hashes,
torch.unsqueeze(hash, 0), p=0).flatten(), k=topk)

mask = img_dataset_embed_100[top2k_indices.tolist()]


img_hash_embeds_pdt = torch.matmul(mask, embed_100)

_, top3k_indices =torch.topk(img_hash_embeds_pdt.flatten(),k=topk)
```

Time [N =50k, d = 100, h = 48, k = 250] = 0.0044 sec
(*faster!*)

Computes hamming distance[Θ(Nh)]
[binary embeddings]

Keeps only ***top 2k*** relevant images

Compute cosine similarity [Θ(kd)]

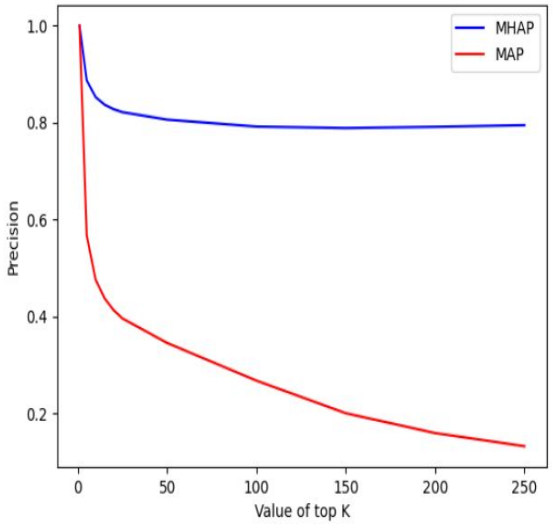
Extract Top k images [Θ(Nh + 2kd)]

```
Model inference and pre-process time : 0.20836884800007738
Query image label :  apple
Time to compute similarity between 100-d embeddings of all images : 0.00033075700002882513

Top similar images based on 100-d embeddings
Similarity:0.9999996423721313 Label:apple
Similarity:0.9990593791007996 Label:apple
Similarity:0.9989938139915466 Label:apple
Similarity:0.9989478588104248 Label:apple
Similarity:0.9987916946411133 Label:apple
Similarity:0.9987280368804932 Label:apple
Similarity:0.9985365867614746 Label:apple
Similarity:0.998501181602478 Label:apple
Similarity:0.9984686374664307 Label:pear
Similarity:0.9984617233276367 Label:apple
Similarity:0.9984349608421326 Label:apple
Similarity:0.9983947277069092 Label:apple
Similarity:0.9983585476875305 Label:apple
Similarity:0.9983342885971069 Label:apple
Similarity:0.9983280897140503 Label:apple
Similarity:0.9981845617294312 Label:apple
Similarity:0.9981720447540283 Label:pear
Similarity:0.9981452226638794 Label:apple
Similarity:0.9981366991996765 Label:apple
Similarity:0.9980074167251587 Label:apple
Similarity:0.9979562759399414 Label:apple
Similarity:0.9979217648506165 Label:apple
Similarity:0.9978954195976257 Label:apple
Similarity:0.9978566765785217 Label:apple
Similarity:0.9978384971618652 Label:apple
Similarity:0.9978347420692444 Label:apple
Similarity:0.9977737665176392 Label:apple
```

semantically similar image

Retrieval Results



Method	mAP @1	mAP @5	mAP@20	mAP@100	mAP@250
Algo-2	1	0.56	0.41	0.26	0.13
Hashes+Algo-2	1	0.56	0.40	0.25	0.12

Method	mAHP @1	mAHP @5	mAHP @20	mAHP@100	mAHP@250
Algo-2	1	0.85	0.82	0.79	0.79
Hashes+Algo-2	1	0.85	0.82	0.78	0.77

Average inference time: 0.2237 sec [time to get embeddings, hashes from model]

Mean Balanced classification accuracy by *Algo-2* of 250 retrieved images : 76.89%

Mean Balanced classification accuracy by *Hashes+Algo-2* of 250 retrieved images : 76.48%

Analysis of Binary hashes method

- Complexity of $M_{N\times d} \times V_{d\times 1}$: $O(Nd)$
- Complexity of $\sum_{ones}(M_{N\times h} \oplus V'_{h\times 1})$: $O(Nh)$
- **AP@1 and AHP@1 almost 1** => Ranking mechanism effective in retrieving ***topmost similar image*** accurately.
- Though mAHP and mAP precision scores are affected, ***compensation*** is decrease in retrieval time achieved. [decrease in precision values small (~1-2 %)]
- *Algorithm 2 + hashes* is expected to perform ***much more better*** than *Algorithm 2* in ***retrieval time*** when retrieving images from a ***larger corpus or retrieving a set of images [diff $\propto$ N(d-h)]***, due to the difference in the sizes of embeddings and ***smaller constant factor*** in computing hamming distance.
- ***Another experiment*** : Provide the platform to experts and allow them to provide a ***satisfaction score*** based on ***relevance*** and ***retrieval time***.

Further thoughts...

Instead of Binary hashes, we use PCA on the 100-d embeddings to enforce dimensionality reduction. Experiment is ongoing.

Current results:

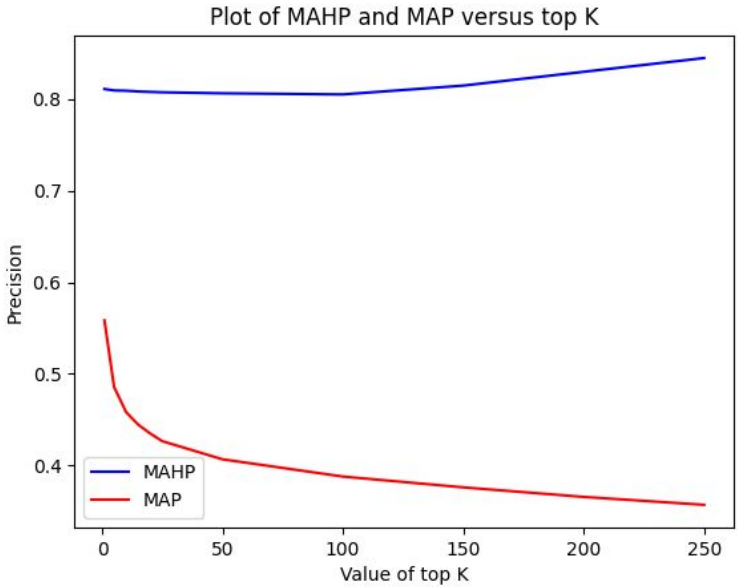


Image Word Associations Model (IWAN)

- Key idea: Construct a graph consisting of image nodes and word nodes, the image node with shortest path from query image reflects most similar image

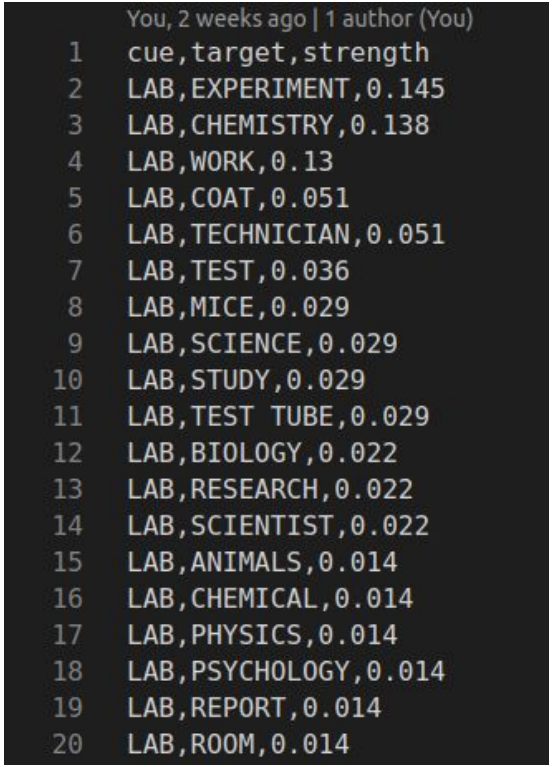
- Pretrained model ResNet-50 predicts word associations for an image and outputs ordered list of words sorted by association strengths.

- Local database of word associations containing top semantically associated words for each word w and the strength of these associations.

- Construct a graph connecting images and words with edge weights corresponding to inverse of association strengths.

Word Association Database

- Local database of word associations constructed from University of South Florida Free Association Norms.
- At least 4 associated words for an existing query word with strengths of the associations.
- Data put inside a PostgreSQL database so that more data from other sources can be later incorporated with the existing one.

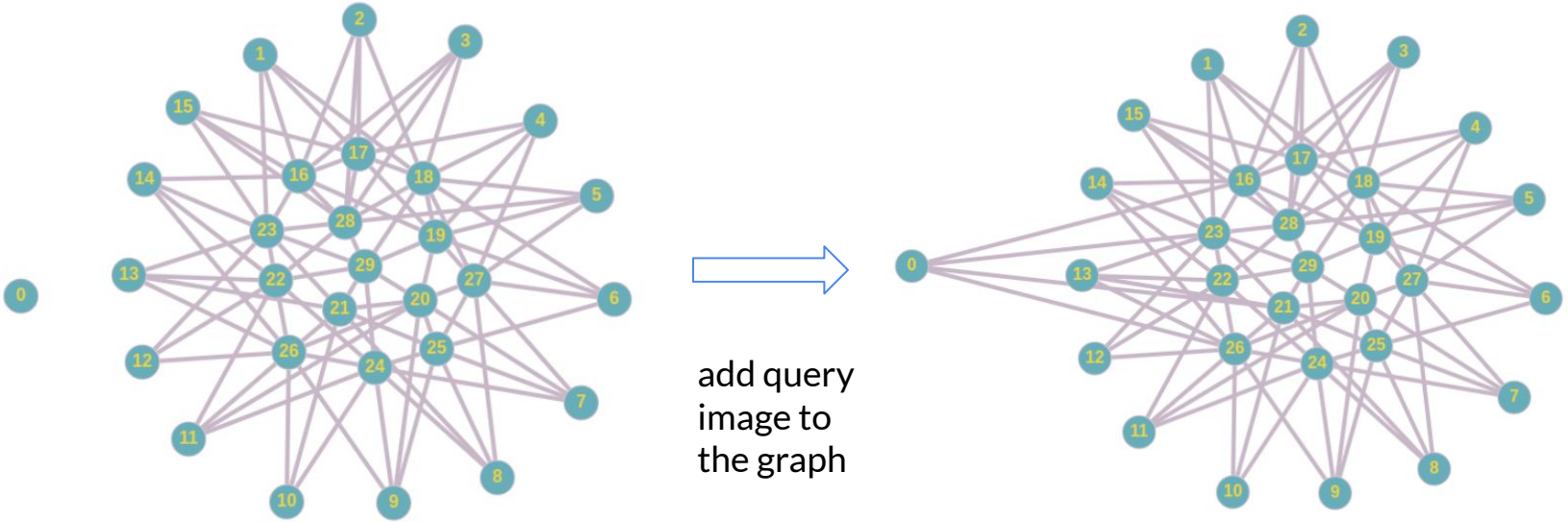


Model Construction

- Graph consists of image nodes and word nodes. Each image node is associated with topIWK (1 <= topIWK <= 10, we take topIWK = 4) word nodes. Each word node is further associated with at least topWWK (topWWK = 4) word nodes.
- Suppose n_images = Number of image nodes, n_words = Number of word nodes. Define n = n_images + n_words + 1.
- Node id 0 reserved for query node. Image nodes are numbered from 1 to n_images whereas word nodes are numbered from n_images + 1 to n_images + n_words.
- If there is an associations between two nodes, there is an edge between them with edge weight as inverse of the association strength.

Retrieval

- A query image is passed through the pretrained ResNet-50 model to get the topIWK associated words.
- The query image node gets the node id 0 and the edges are constructed between this node and topIWK existing word nodes.
- The graph becomes connected.
- A single-source shortest path algorithm (we use Dijkstra's algorithm) from query node 0 to image nodes is performed and the topK image nodes which are nearest to the query node are retrieved.


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
add query image to the graph

Analysis

Metric	Value
Time (no overhead)	0.50s
Time (with overhead)	2.67s
mAP@1	52.07%
mAP@5	53.04%
mAP@10	52.57%
mAP@15	52.09%
mAP@20	51.70%

Performance Metrics for all 10,000 test images

Metric	Value
mAHP@1	59.93%
mAHP@5	60.29%
mAHP@10	59.88%
mAHP@15	59.64%
mAHP@20	59.42%

- Overhead is due to query image undergoing cubic interpolation from 32x32 to 224x224, so that it can be input into the ResNet-50 model. Processing by the ResNet-50 model also adds on top of it.

- Interpolation also results in less accurate predictions by the ResNet-50 model, partly explaining relatively low values of mAPs and mAHPs.

Experiments

1. Variation of mAP with topIWK
2. Variation of mAHP with topIWK
3. Comparison of mAP@20 and mAHP@20
4. Variation of Retrieval Time with topIWK

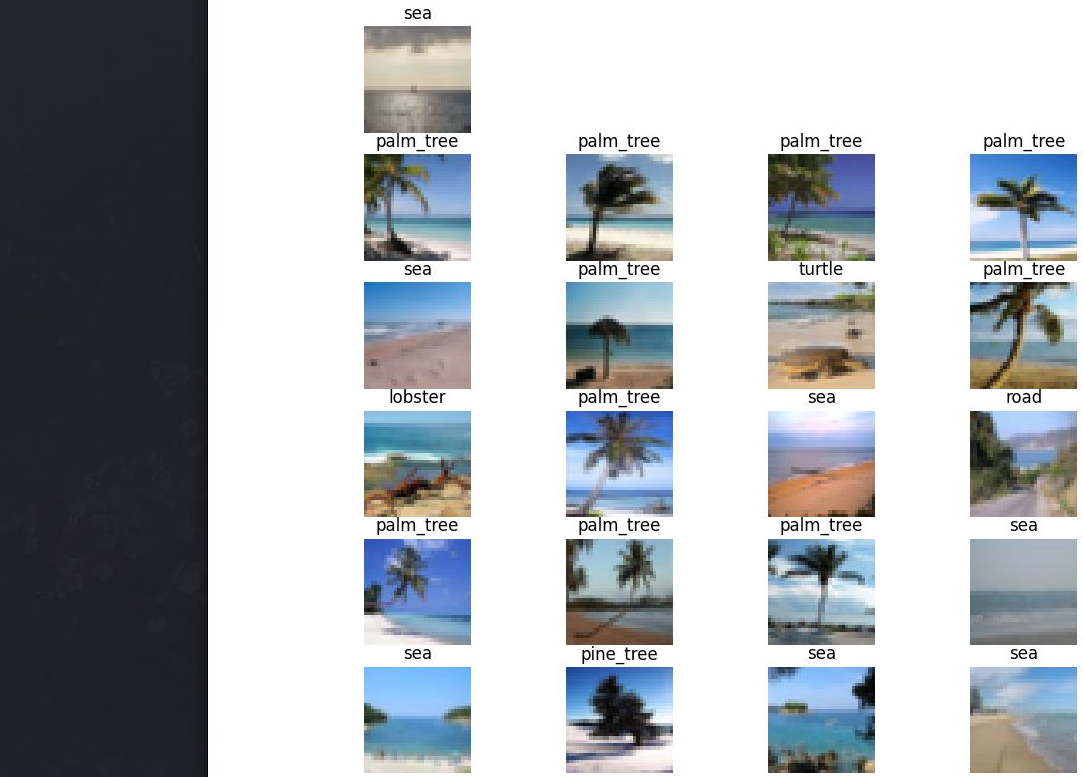
Alacritty

4	41899	palm_tree	trees
5	30887	sea	large_natural_outdoor_scenes
6	25780	palm_tree	trees
7	24762	turtle	reptiles
8	40271	palm_tree	trees
9	49371	lobster	non-insect_invertebrates
10	2843	palm_tree	trees
11	20455	sea	large_natural_outdoor_scenes
12	29548	road	large_man-made_outdoor_things
13	5429	palm_tree	trees
14	6766	palm_tree	trees
15	31772	palm_tree	trees
16	25140	sea	large_natural_outdoor_scenes
17	43091	sea	large_natural_outdoor_scenes
18	29870	pine_tree	trees
19	41251	sea	large_natural_outdoor_scenes
20	30972	sea	large_natural_outdoor_scenes

Retrieval Statistics

Metric	Value
Retrieval Time (without overhead)	0.49s
Retrieval Time (with overhead)	2.58s
P@1	0.00%
P@5	20.00%
P@10	10.00%
P@15	13.33%
P@20	30.00%
HP@1	0.00%
HP@5	20.00%
HP@10	10.00%
HP@15	13.33%
HP@20	30.00%

Figure 1



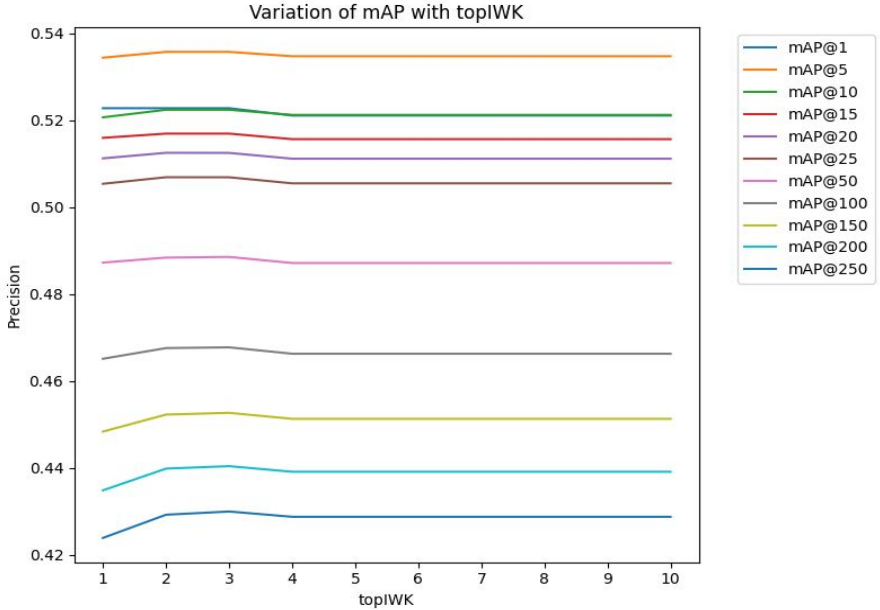


Figure 1 - Variation of mAP with topIWK

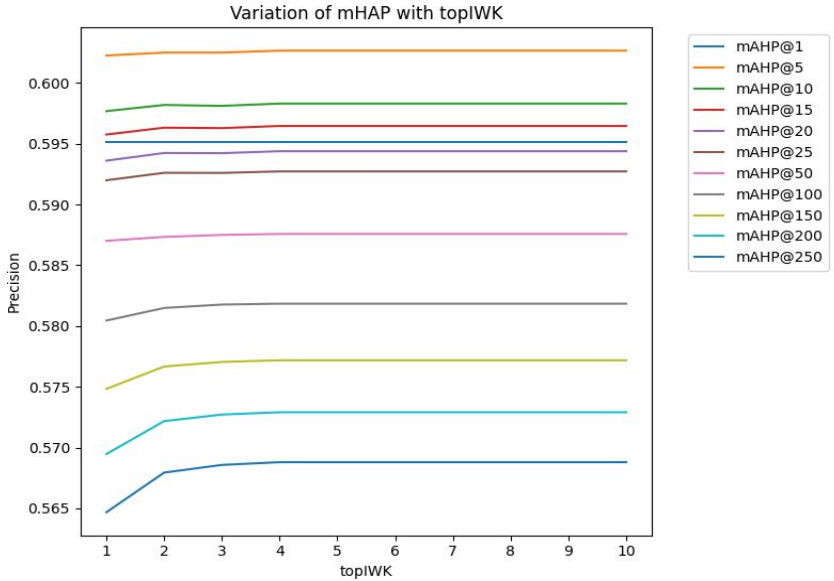


Figure 2 - Variation of mHAP with topIWK

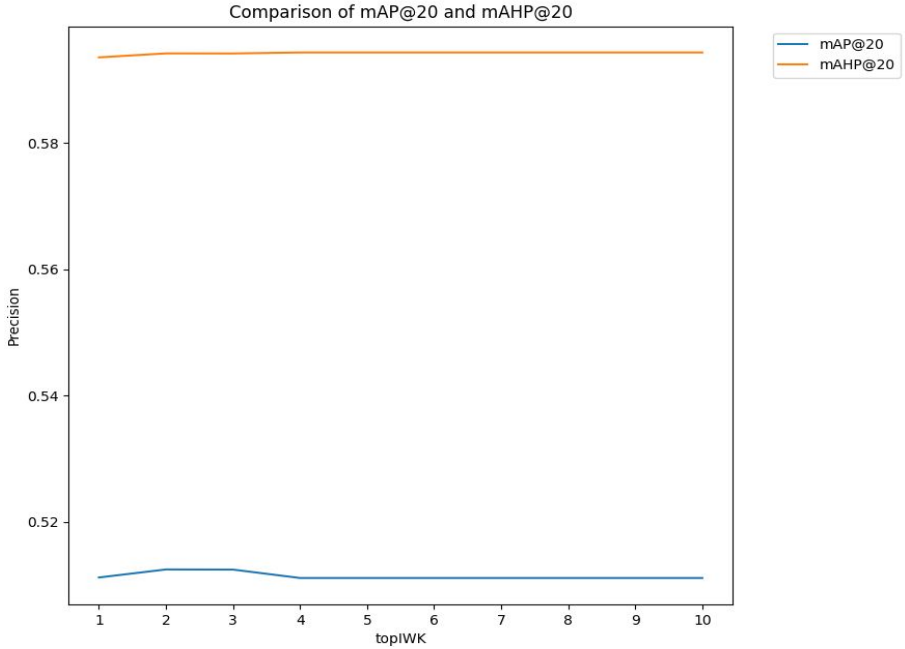


Figure 3 - Comparison of mAP@20 and mAHP@20

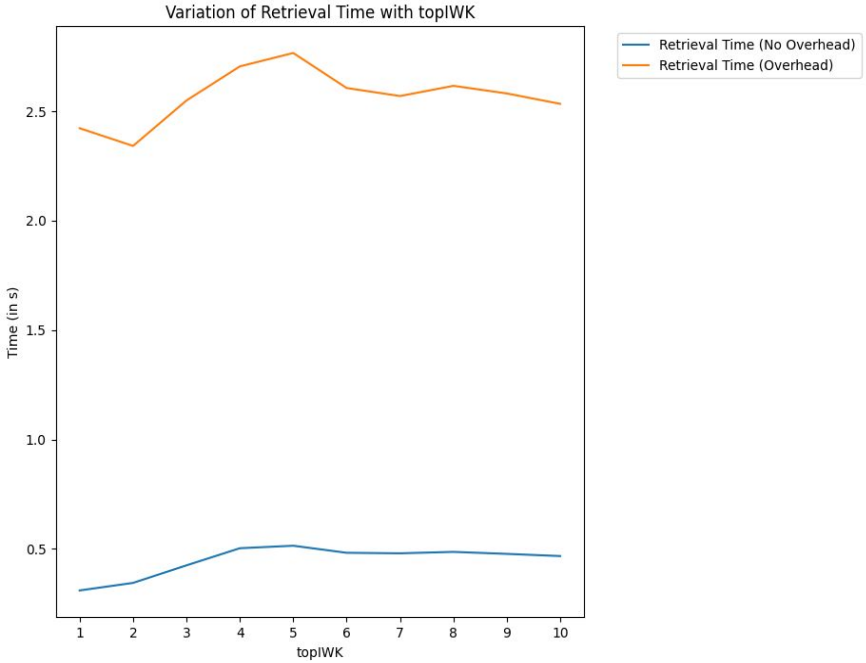


Figure 4 - Variation of Retrieval Time with topIWK

Work Division

Name	Experiments and Ideation
Rounak Saha	Literature survey and ideation of Algo 2 and Algo 2 + hashes as a combination of methods proposed in different papers, coding the NN architecture and fine tuning it, optimised code to improve retrieval time, report and ppt
Swarup Padhi	Conducted all experiments (precision,hierarchical precision,balanced accuracy,retrieval time) related to Algo 2 and Algo 2 + hashes, optimised code to improve retrieval time and designed query interface, report and ppt.
Ritwik Ranjan Mallik	Created Postgres database and calculated shortest path in IWAN graph by using Dijkstra's algorithm to find topK nearest images. Analysis of IWAN model performance metrics: Variation of mAP and mAHP with topIWK, Comparison of mAP@20 and mAHP@20, Variation of Retrieval Time with topIWK, report and ppt.
Saptarshi De Chaudhury	Constructed graph for IWAN using ResNet-50 and word association database. Using PCA for dimensionality reduction of 100-d embeddings. Plotted graphs for the mAHP and mAP vs value of top K, report and ppt.