Fast Image Retrieval using Binary Hashes and Network of Word Associations

Rounak Saha **Swarup Padhi** **Ritwik Ranjan Mallik** **Saptarshi De Chaudhury**

1 Problem statement and motivation

This project addresses the problem of content-based image retrieval, which involves searching for digital images in large databases based on their content rather than textual descriptions or metadata. Although visual similarity is often used to cluster similar images in feature space, it does not necessarily correspond to semantic similarity. Moreover, using a classification objective to extract image features for content-based image retrieval does not guarantee a high distance between unrelated classes, leading to incorrect nearest neighbors for certain images. To overcome these issues, we propose an extension of hierarchy-based semantic embeddings for nearest neighbor search using the CIFAR-100 dataset. We explore different indexing schemes to make query processing faster while maintaining comparable retrieval metrics. Our approach involves searching for k-nearest neighbors in a high-dimensional space, given image embeddings and a black box model to convert images to embeddings. We examine two indexing strategies, one using binary hash codes learned via deep learning and the other using a network of word associations.

2 Related Work

The common approach of using categorical cross-entropy loss and softmax activation to train CNN classifiers results in features that are well-separable but not necessarily discriminative. Barz et al. [1] proposed learning hierarchy based embeddings of image classes such that the pairwise dot products of their embeddings equal their semantic similarities. The paper proposes a network based metric to measure semantic similarity between entities (using WordNet[2]) and an algorithm for learning the embeddings that conform to those similarity values. To address the problem of efficiency of retrieving k nearest neighbors in a high dimensional space for CBIR, researchers have explored Approximate Nearest Neighbor(ANN) search techniques based on hashing based methods [6,7,8,9,10,11] instead of linear search. Lin et al. [3] proposes to learn binary hash codes from the activation of a hidden layer of CNN trained for classification tasks. In our work, we try to explore a combination of the above two methods and use the class embeddings of CIFAR-100 dataset made available by Barz et al. [1] to train a neural network model that predicts class embedding instead of softmax probability and adopt the strategy of learning binary codes from hidden layer as shown by Lin et al. [3] We then perform evaluation of retrieval metrics including time.

The concepts of graph theory can be used to reason about our word associations graph and come up with ideas to improve upon the state-of-the art Content-Based Image Retrieval systems. Palumbo et al. [4] propose to represent a knowledge base of semantic concepts as a complex network whose topology arises from free conceptual associations. Images are anchored to relevant semantic concepts through an annotation process and similarity is computed following the related paths in the complex network. In this context, the problem of retrieving semantically similar images given a query image reduces to finding single-source shortest paths in a graph. We build a system that implements this idea for retrieval of images based on semantic similarity and evaluate its retrieval metrics against its average query retrieval time.

3 Method

We propose two techniques:

3.1 Deep learning hierarchy-based binary hash codes

This CBIR framework has 3 main components:

3.1.1 Estimate class similarities and learning class embeddings

This section involves learning hierarchy based embeddings of image classes such that the pairwise dot products of these embeddings equal their semantic similarities. The intra-class variance of the learned representations must be discriminatively small compared to the inter-class variance. i.e. images of the same class should be tightly clustered together in feature space. However neighboring clusters in the feature space should not be completely semantically unrelated. On a dataset of m classes and n images, this task can be subdivided into two parts: (1) Estimate pairwise semantic similarities between classes and construct the class similarity matrix $[M]_{m\times m}$ (2) Learn class embedding matrix $[C]_{d\times m}$ such that C_i equals embedding of the i-th class and for all pairs of classes (p, q), $cosine(C_p, C_q) \approx M_{p,q}$

3.1.2 Learning a transformation of input images to the embedding space and a lower dimensional latent hash space

The second part consists of learning two transformations, namely $\mathcal{F} : X \longrightarrow \mathbb{R}^d$ and $\mathcal{H} : X \longrightarrow \{0,1\}^h$ from the space of input images X to hierarchy-based semantic embedding space and latent hash space respectively where $h << d$. This is achieved by supervised training of a deep neural network model that should take in as input an instance of the image space and outputs vector embeddings and hashes of dimensions d and h respectively. Given an image classification dataset $D = \{(image_i, classlabel_i)|i = 0, 1, 2, ..., n-1\}$ of m classes and n images, the training can be carried out in the following way: (1) construct a modified dataset D' from D such that for each $image$ the $classlabel$ is replaced by a vector $C_{classlabel}$ where $classlabel$ is the index of the class that $image$ belongs to and C is the same class embedding matrix we pre-computed in the first module of the CBIR framework. (2) add a decoder MLP to a standard CNN model and the final layer of the MLP would consist of d neurons that would try to approximate the class embedding (3) train the model with a objective that optimizes the cosine loss between the learned representations and the class embeddings (4) use the binarized form of a low dimensional hidden layer as hashes, note that the hashes are obtained as a byproduct of the training carried out as explained above.

The architecture we have used consists of the standard AlexNet architecture which converts an input image of dimensions (224,224,3) to a 4096-vector (we strip the final classification layer of the AlexNet and use the rest of it as the encoder) and a bilayer MLP of 48 and 100 neurons respectively as the decoder, the decoder output thus is a 100-d vector which approximates the class embedding. The hidden layer of 48 neurons is followed by a sigmoid activation layer which restricts the activations between [0,1] so that they can be used suitably as hashes as we describe later. We train the model on CIFAR-100 dataset 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:

$$\mathcal{L}(x, y) = 1 - cos(x, y)$$

3.1.3 Hierarchical deep search using binary hashes followed by embeddings

Having learned the transformations, we pre-compute d-dimensional embeddings and h-dimensional hashes for each image of the database of size N in the form of the image embedding matrix $[V]_{d\times N}$ and image hash matrix $[H]_{h\times N}$. The hashes are then binarized based on a threshold.

$$H_i = \begin{cases} 1 & \text{if } Out_i > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

When a query image q comes in, we apply the same transformation and extract the embedding v_q and hash h_q (followed by binarization) by feeding q as input to the model. With these entities in hand we describe how retrieval of top k images is carried out:

(1) We first describe how the search is carried out efficiently in the absence of any binary hashes. We first present Algorithm 1 which naively goes over each embedding one by one and calculates the cosine similarities. However, we note that this entire algorithm can be executed much more efficiently using a single matrix multiplication followed by search of k-th largest element. Extremely efficient implementations of these fundamental operations based on low-level optimizations and parallel processing are made available by standard libraries like Numpy and Pytorch. This is demostrated in Algorithm 2. The function $topk(k)$ returns the values and indices of the k largest elements in the vector. Viewing the search as matrix manipulation made possible due to using cosine similarity as

Algorithm 1 Naive Retrieval Function

```
1: function NaiveRetrieval($v_q$, $[V]_{d\times N}$, $k$)
2:     similarities ← List< simval, id >
3:     for $i \leftarrow 0$ to $N-1$ do
4:         sim ← cosine_similarity($v_q$, $V_i$)
5:         similarities.append(⟨sim, i⟩)
6:     end for
7:     sort(similarities, order = descending)
8:     results ← List()
9:     for $j \leftarrow 0$ to $k-1$ do
10:        results.append(similarities[j].id)
11:    end for
12:    return results
13: end function
```

Algorithm 2 Retrieval by Matrix Multiplication

```
1: function RetrievalByMatrixMult($v_q$, $[V]_{d\times N}$, $k$)
2:     similarities ← $V^T v_q$
3:     values, indices ← similarities.topk($k$)
4:     return indices
5: end function
```

similarity metric constitutes the first level of optimization that we achieve over naive search.
(2) We now describe how the hashes are used to speed up the query retrieval process. From the complete database of N images, we shortlist a pool of N' (N' is $2k$ in our implementation) images (N' being considerably smaller than N) whose hashes have the lowest Hamming distance from the hash of the query image. For each candidate we compute similarity using d-dimensional embeddings and retrieve the top k results using Algorithm 2. The dimension of hashes is much shorter than the actual embeddings, also Hamming distance calculation is easier since it only involves simple bitwise operations. Since the initial shortlisting reduces the space for exhaustive search considerably, we should get a speed up in retrieval time for very high dimensional embeddings or massive datasets, but at the cost of approximate results.

3.2 Complex Network of Free Word Associations

We use IWAN (Image Word Associations Network) as the model name. This CBIR system has the following components:

3.2.1 Estimating image-word and word-word associations

We have a database D of N images and a vocabulary of words of size M. For estimating image-word associations, we have at our disposal a pre-trained model $\mathcal{M}$ which, given an image I, outputs an ordered list of words sorted according to the confidence scores of them being associated with I. We also have a database of word associations which contains, for each word w, the top t words which are semantically associated with w and the strength of these respective associations. We use the Resnet-50 model as $\mathcal{M}$, pre-trained for the task of predicting word associations given an input image. We use the University of South Florida Free Association Norms [5] as the database of word associations which contains 72177 associations, we take $t = 10$

3.2.2 Constructing the Image Word Associations Network

In the IWAN, each image is associated with some $topIWK$ number of words. Each of these words are associated with some $topWWK$ words. These associations are not necessarily disjoint for two images. In this way we determine the set of all words in the IWAN. With $topIWK = 4$ and $topWWK = 4$, the number of words in the IWAN comes out to be 1800. The set of words are sorted and assigned ids starting from 0.
The IWAN contains two types of nodes, image nodes (N) and word nodes (M). In addition, there is a query node which is a special kind of image node. The graph therefore has $n = N + M + 1$ vertices. They are numbered from 0 to $N + M$, both inclusive. The query node gets the id 0, whereas the image nodes get ids from 1 to N, both inclusive. Finally, the id from $N + 1$ to $N + M$ are assigned to the words. The IWAN is an undirected weighted graph. There is an edge between an image node and a word node if and only if the word is associated with the image (obtained from the ResNet-50 model) and the weight of this connection is the inverse of the strength of the association of the word with the image. Similarly, there is an edge between two words if and only if they have some association score as retrieved from the database and once again, the weight of this edge is the inverse of the strength of the association. Taking the inverse and setting it as edge weight ensures that the stronger the association, the closer the nodes are with each other. The query node with id 0 does not have any

association with any word and therefore has degree 0. This undirected weighted graph with two connected components is essentially the IWAN model.

3.2.3 CBIR using IWAN

In this section, the query node is merged with the rest of the network and the images are retrieved using a shortest path metric. The query retrieval overhead thus is the cost of adding the edges of the query node and shortest path calculation, note that the rest of the graph need not be constructed every time a query is performed. The procedure is outlined in Algorithm 4. The $find\_node$ function finds the id of a word node given the word name and $Dijkstra(source)$ returns the vector containing shortest paths of all nodes from source.

Algorithm 3 IWAN Retrieval

```
 1: function IWAN_RETRIEVAL($IWAN$, $I_q$, $k$)
 2:     $IWAN.nodes_0 \leftarrow I_q$
 3:     $L \leftarrow \mathcal{M}(IWAN.nodes_0)$
 4:     for $i = 0$ to $topIWK - 1$ do
 5:         $node\_id \leftarrow find\_node(L_i)$
 6:         $add\_edge(0, node\_id)$
 7:     end for
 8:     $path\_len \leftarrow Dijkstra(source = 0)$
 9:     $value, indices \leftarrow (-path\_len).topk(k)$
10:     return indices
11: end function
```

3.3 Experiments

We use the CIFAR-100 test images and evaluate the performance of query retrieval of both methods on following metrics: average query retrieval time, mean average precision mAP@k (k=1,5,10,15,250) and mean hierarchical precision mAHP@k (k=1,5,10,15,250).

3.4 Results and Analysis

The results from Algorithm 2 and Algorithm 2 + hashes is given in Appendix (Table 1 and 2). In Algorithm 2, we multiply an $N \times d$ matrix with a d-dimensional embedding vector. In contrast, in Algorithm 2 + hashes, we have computed hashes of dimension h with the aim of improving the query retrieval time without significantly affecting hierarchical precision. Finding the hamming distance between rows of an $N \times h$ dimensional matrix with an h-dimensional vector should result in lower computation time than a multiplication of an $N \times d$ dimensional matrix with a d-dimensional vector, where $d > h$.We observed that it took an average of 4.6 ms by Algorithm 2 and 4.4 ms sec by Algorithm 2 + hashes to retrieve the top 250 images from the dataset. Though mAHP and mAP precision scores are affected, compensation is decrease in retrieval time achieved. [decrease in precision values small ($\sim 1 - 2\%$)]. However, we are retrieving images corresponding to a single query and from a small corpus. We expect that if we retrieve images from a larger corpus than CIFAR -100, our Algorithm 2 + hashes would perform better than Algorithm 2. Additionally, if we wish to retrieve results for a set of images, it would be even faster since the disparity in the size of embeddings is expected to make Algorithm 2 + hashes perform faster.

We also obtain a very high value of Precision@1 and Hierarchical Precision@1 using both Algorithm 2 and Algorithm 2 + hashes. This suggests that our ranking mechanism effectively retrieves the topmost similar image accurately. We intend to evaluate our method by providing the platform to experts and allowing them to provide a satisfaction score based on relevance and retrieval time.

The results from the IWAN model is given in Appendix (Table 3 and 4). It is evident that whereas the actual Dijkstra's Algorithm for single-source shortest paths takes 0.5s on average, the retrieval time for the labels is 2.5s on average (tested using CPU: Intel i5-10300H and GPU: Nvidia RTX-2060). This overhead is due to every CIFAR-100 32x32 test image undergoing cubic interpolation to produce a 224x224 image, a format suitable for the ResNet-50 model. Moreover, the ResNet-50 model adds its own overhead on top of it as it takes in the input image and produces the associated words along with their strengths. If the dataset consisted of 224x224 images and the model more compact, retrieval would have been faster. After the algorithm has run, it has in fact found shortest paths to all the image nodes, out of which we are retrieving only the topK images. This means that the retrieval time is independent of the value of topK. Average Precisions (P@5, P@10, P@15, P@20) are binary precisions wherein the relevance score is 1 if the fine labels match, otherwise 0. Hierarchical Average Precisions (HP@5, HP@10, HP@15, HP@20), on the other hand, uses dot products of the embeddings of the labels and therefore, HP@K

>= P@K in most cases. The relatively low values for the mean of these quantities (mAP@K's and mAHP@K's) is due to the cubic interpolation. Also different values of topIWK and topWWK would lead to more accurate results. We present an experiment to observe the impact of topIWK on the retrieval metrics in the Appendix.

Other than finding images nodes which are near the query node, different measures can be considered as well. For example, we can consider the sum of all possible path costs from query node to a given image node and less is this value, more is the corresponding training image semantically similar to the query image.

The heirarchy based embeddings used to evaluate the model performance are also not fully perfect. For example, the fine label "sea" and "palm_tree" have similarity score 0 as per both binary and hierarchical precision. However, an image of a sea is semantically similar to the image of a palm tree which is demonstrated by the IWAN model.

3.5 References

[1] Hierarchy-based Image Embeddings for Semantic Image Retrieval: Bjorn Barz. Joachim Denzler [2] C. Fellbaum. WordNet. Wiley Online Library, 1998. [3] Deep Learning of Binary Hash Codes for Fast Image Retrieval, Kevin Lin, Huei-Fang Yang, Jen-Hao Hsiao, Chu-Song Chen [4] Semantic Similarity Between Images: A Novel Approach Based on a Complex Network of Free Word Associations: Enrico Palumbo and Walter Allasia [5] The University of South Florida Free Association Norms [6] A. Gionis, P. Indyk, R. Motwani, et al. Similarity search in high dimensions via hashing. In VLDB, volume 99, pages 518–529, 1999 [7] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In Proc. NIPS, pages 1753–1760, 2009 [8] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In Proc. CVPR, pages 2074–2081, 2012. [9] M. Norouzi and D. M. Blei. Minimal loss hashing for compact binary codes. In Proc. ICML, pages 353 360, 2011. [10] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In Proc. NIPS, pages 1042–1050, 2009. [11] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan. Supervised hashing for image retrieval via image representation learning. In Proc. AAAI, 2014

4 Appendix

We present the retrieval metrics for hierarchy-based binary hashing (Algorithm-2 and Hashes + Algorithm-2) in Table 1 and 2, and for IWAN in Table 3 and 4.

Metric	**Value**
Inference Time	220 ms
Retrieval Time	4.67ms
mAP@1	1.0
mAP@5	0.5656
mAP@20	0.4119
mAP@100	0.2664
mAP@250	0.1321
mAHP@1	1.0
mAHP@5	0.8865
mAHP@20	0.8275
mAHP@100	0.7916
mAHP@250	0.7942

Table 1: Test results of Algorithm 2 on all Test Images

Metric	**Value**
Inference Time	220 ms
Retrieval Time	4.48ms
mAP@1	1.0
mAP@5	0.5617
mAP@20	0.4057
mAP@100	0.2559
mAP@250	0.1236
mAHP@1	1.0
mAHP@5	0.8865
mAHP@20	0.8243
mAHP@100	0.7838
mAHP@250	0.7786

Table 2: Test results of Hashes + Algorithm 2 on all Test Images

We perform some more analysis on IWAN, viz. we observe the variation of retrieval metrics by varying the value of $topIWK$

In Figure 1, we plot the variation of mAP with $topIWK$. The precision values are low for lower values of topIWK, becomes a maximum at $topIWK$ = 4 and subsequently decreses and eventually levels out as $topIWK$ increases. For low values of $topIWK$, less paths are available and hence precision decreases. On the other hand, for high values of $topIWK$, many more paths are available some of which may lead to wrong predictions. The

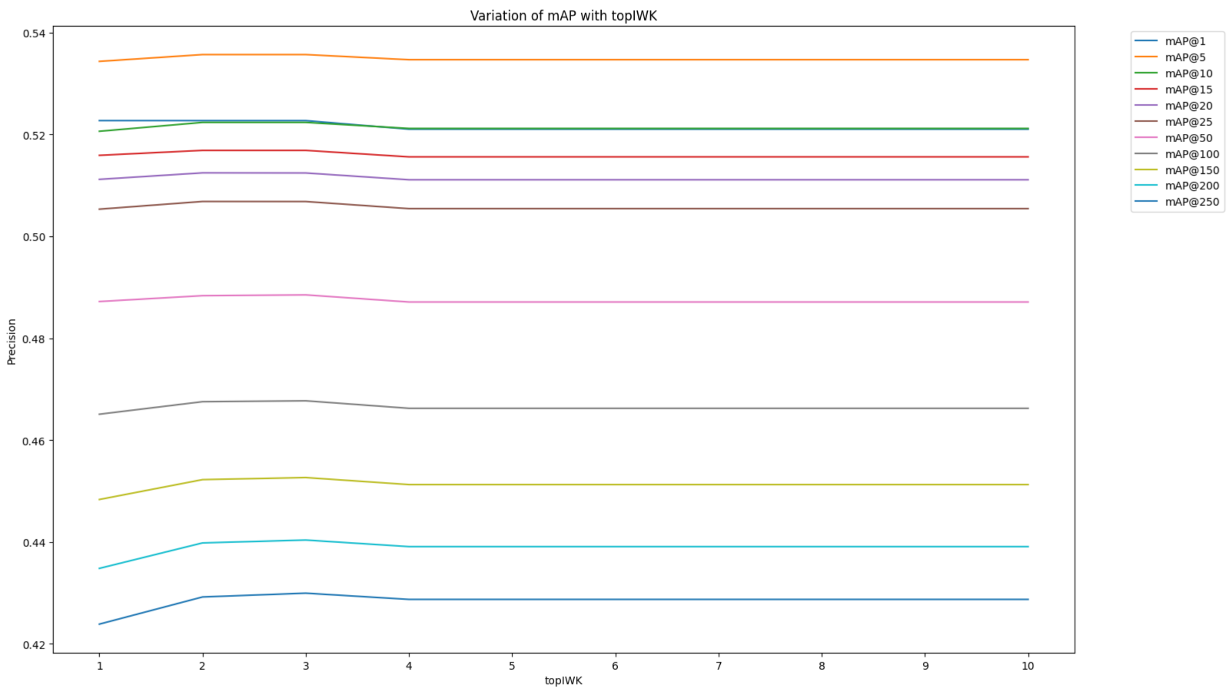

Figure 1: Variation of mAP with topIWK

Metric	**Value**
Retrieval Time (without overhead)	0.42s
Retrieval Time (with overhead)	2.54s
mAP@1	0.5227
mAP@5	0.5356
mAP@20	0.5124
mAP@100	0.4676
mAP@250	0.4299
mAHP@1	0.5227
mAHP@5	0.5346
mAHP@20	0.5124
mAHP@100	0.4676
mAHP@250	0.4299

Table 3: Test results of IWAN model with topIWK=3

Metric	**Value**
Retrieval Time (without overhead)	0.50s
Retrieval Time (with overhead)	2.7s
mAP@1	0.5210
mAP@5	0.5346
mAP@20	0.5111
mAP@100	0.4662
mAP@250	0.4286
mAHP@1	0.5210
mAHP@5	0.5346
mAHP@20	0.5111
mAHP@100	0.4662
mAHP@250	0.4286

Table 4: Test results of IWAN model with topIWK=4

leveling out highlights the fact that the number of words in the vocabulary eventually becomes constant and further increase of $topIWK$ will not add better paths.

In Figure 3, we plot the variation of retrieval time with $topIWK$. Retrieval time with overhead consists of added in time taken by cubic interpolation and predictions by the ResNet-50 model. No definite conclusion can be made about this measure at it depends on external factors. Retrieval time without overhead is low for low values $topIWK$ as the number of words is less. However, it eventually becomes constant because after a certain point, no new words are added to the network.

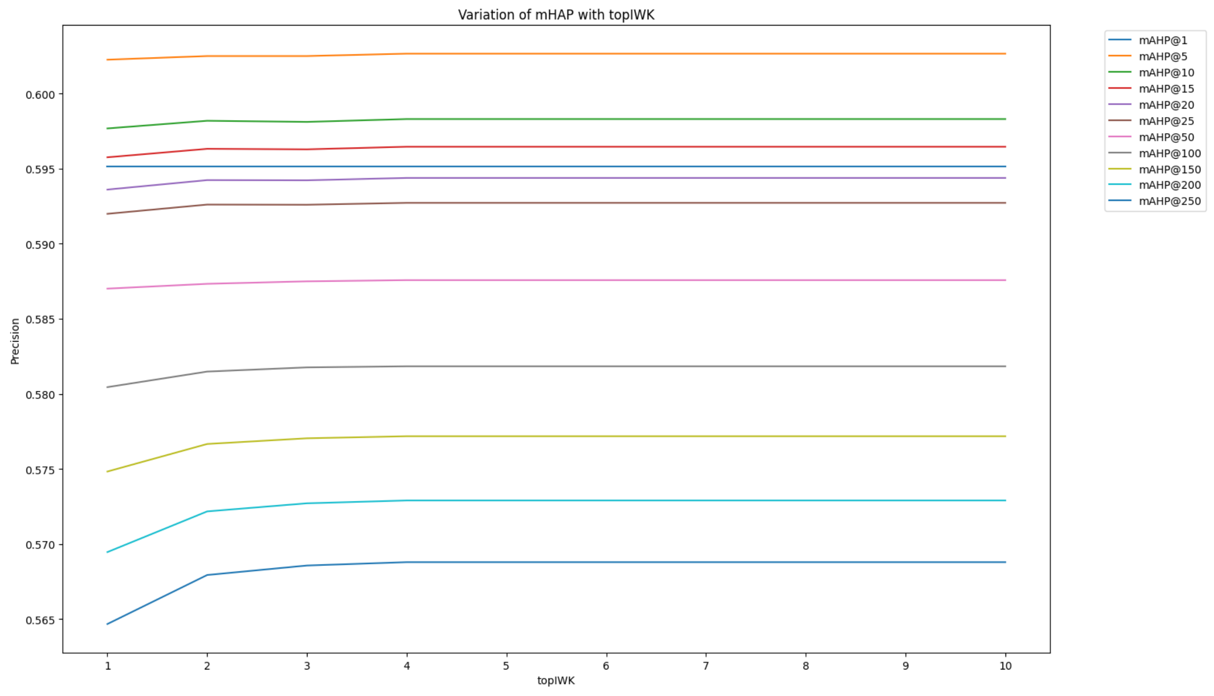

Figure 2: Variation of mAHP with topIWK

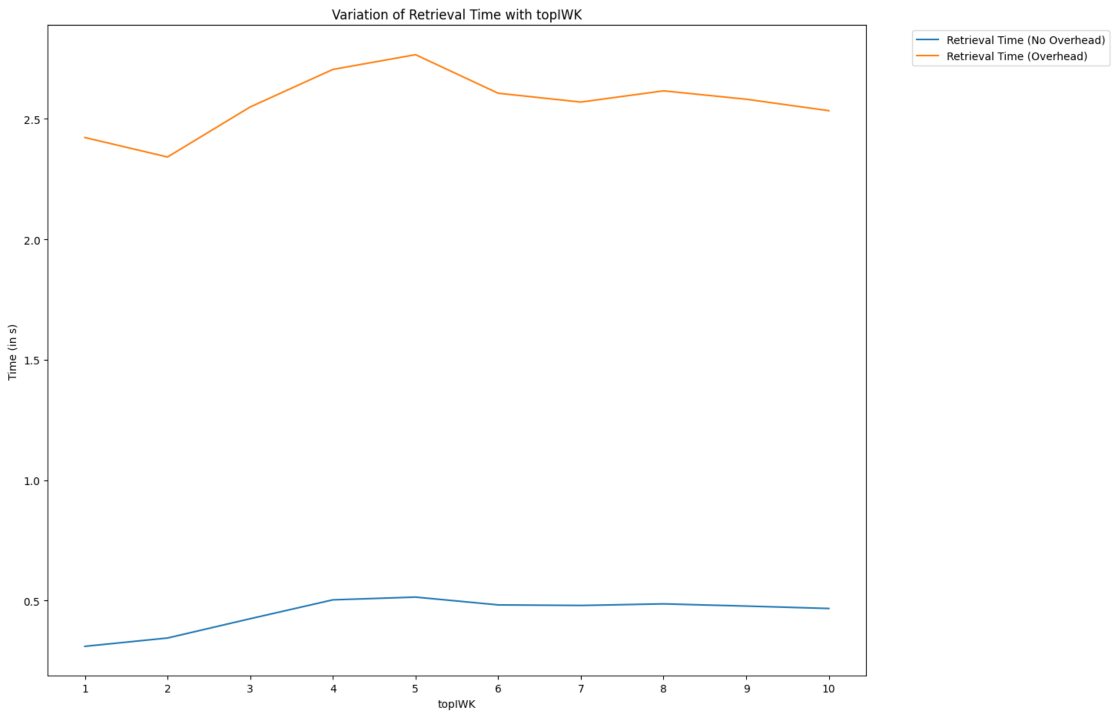

Figure 3: Variation of retrieval time with topIWK

Work Distribution

Names	Tasks
Rounak Saha	Literature survey, 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, optimized code to improve retrieval time, report and ppt
Swarup Padhi	Literature survey, conducted all experiments (precision,hierarchical precision,balanced accuracy,retrieval time) related to Algo 2 and Algo 2 + hashes, optimized code to improve retrieval time and designed query interface, report and ppt.
Ritwik Ranjan Mallik	Literature survey, ideation of the IWAN method, 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	Literature survey, 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.