Image Classification using Hashing Techniques and Relevance Feedback

Group Members

1. Dhruvil Deepakbhai Shah / dshah47@asu.edu
2. Deep Vijaykumar Patel / dpatel94@asu.edu
3. Nihar Samirbhai Patel / npatel69@asu.edu
4. Manikanta Mudara / mmudara@asu.edu
5. Pramukh Belam / pbelam@asu.edu
6. Vijay Aravindh Viswanathan / vviswa19@asu.edu

Abstract

During this phase, images are labeled image-X-Y-Z.png, where – X (cc, con, emboss, jitter, neg, noise1, noise2, original, poster, rot, smooth, stipple) specifies the image type, Y denotes the subject ID in range [1-40], and Z denotes the image sample ID [1-10]. The three feature models and similarity/distance functions generated in the previous phase are used in this phase's tasks. Various dimensionality reduction techniques like Principal Component Analysis, Latent Dirichlet Allocation, Singular Value Decomposition, K-means clustering are applied to the extracted features. Then we use a classifier such as Support Vector Machine, Decision Tree, and Personalized Pagerank to classify images. Another approach uses Hashing Techniques like Locality Sensitive Hashing and Vector Approximation files along with K-nearest neighbors to extract similar images.

Keywords

SVM, Decision Tree, PPR, LSH, VA, Relevance feedback, SVD, Color Moments, ELBP, HOG

Introduction

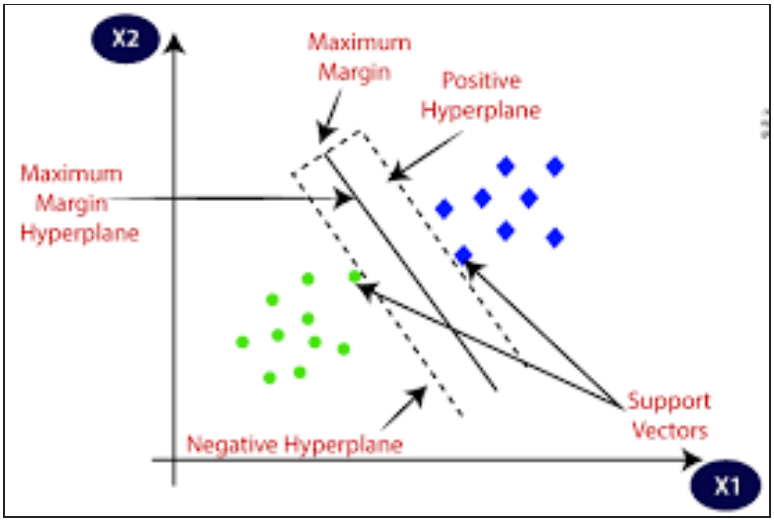
The dataset contains images of 12 classes. Each class contains 40 subjects, and each subject has 10 various images. The dimensions of the images are 64 x 64. Dimensionality reduction techniques cannot be applied if the length of feature vectors is not uniform. To tackle this issue, we dropped some images from some classes and got 360 images for each class type. For each image, Color Moments returns a feature vector of length 192 (8x8x3), Extended Local Binary Pattern return a feature vector of length 256 (0-255), Histogram of Oriented Gradients returns a feature vector of length 1764 (49x36). In the dimensionality reduction functions, we give the data matrix of NxM (where N = number of objects, M = number of features), and it returns the Nxk (where N = number of objects, k = number of reduced dimensions) dimension matrix.

Terminology

This section highlights the basic terminologies used throughout the implementation of the tasks of this project phase.

Support Vector Machines:

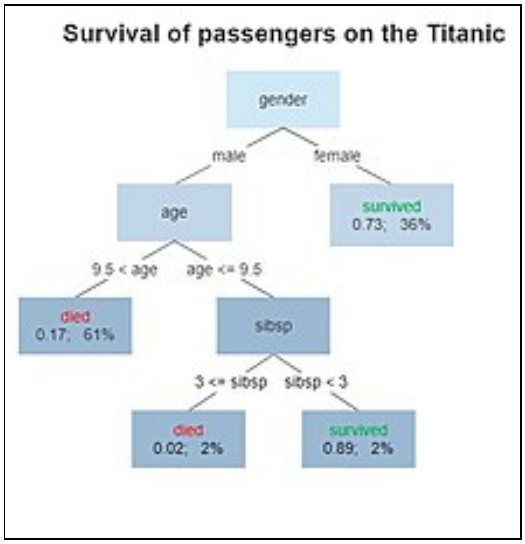
Support vector machines is a supervised machine learning algorithm which analyzes data for classification and regression analysis. It classifies data into one of two categories. An SVM generates a map of the sorted data with the borders between the two as far apart as possible. SVMs are highly used in text categorization, Object classification, handwriting recognition and in the sciences. As in all other supervised learning algorithms, SVM too requires labeled data to be trained as input. SVM is trained with a series of data already classified into two categories, building the model as it is initially trained. SVM algorithm then determines which category a new data point belongs in. For multi-class data, SVM divides data into two classes, the main class and all other classes. Support Vector Machines are based on the concept of decision planes that define decision boundaries. A decision plane is one that separates between a set of objects having different class labels.



Decision Tree:

Decision Tree is a supervised learning algorithm, which splits the data into multiple buckets based on if condition. So at each node the data is split recursively till the leaf nodes are reached. A decision tree contains three types of nodes: Decision Nodes, Chance Nodes, and End Nodes. For every decision node, special care is taken to select the variable which provides highest entropy for the remaining data. A decision tree can be linearized into decision rules where the outcome is the contents of the leaf node, and the conditions along the path form a conjunction in the if clause. In general, the rules have the form:

if condition1 and condition2 and condition3 then outcome.



Personalized Page Rank:

The Pagerank algorithm is used to determine a webpage's quality or importance. The Pagerank of a webpage is determined by counting the number of web pages that connect to it and the number of outgoing links from each of those web pages. In general, Pagerank can be deduced by examining a person who visits a specific webpage. The user jumps to any random webpage with probability 1 - Beta, and with probability Beta, he jumps to one of the web pages pointed by the current webpage.

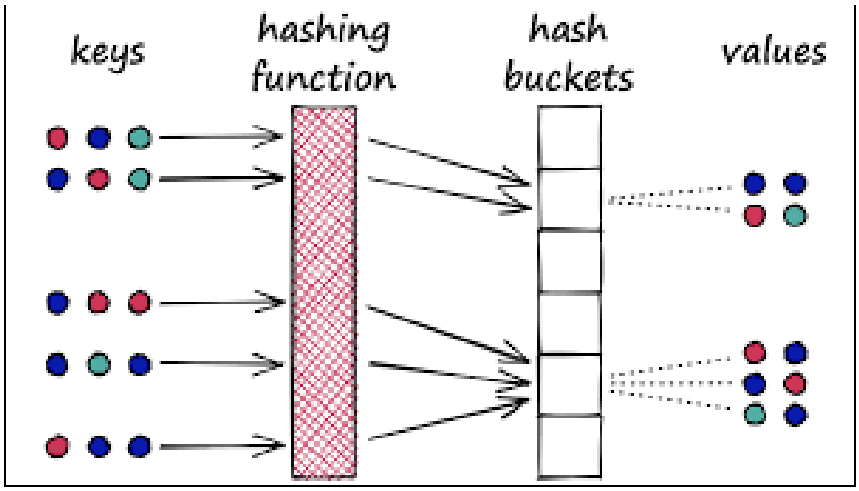
$$PR(t + 1) = \beta * (T X PR(t)) + (1 - \beta) * s$$

Where T denotes the Transition Matrix, Beta denotes the likelihood that a user will jump to one of the nodes in the current node's outbound set, PR(t) denotes the set of Page Rank of all nodes at iteration t, and s denotes the seed matrix.

Locality Sensitive Hashing:

Locality Sensitive Hashing is similar to a K Nearest Neighbour search, with the exception that it is less computationally expensive. While in KNN we compare every data point to look for comparable objects, with LSH we just compare data points within the same bucket. The buckets are set up so that each one contains only data points that are spatially related. LSH generates hash codes for data points in such a way that hash codes for related data points are the same. It essentially uses a code to label a subspace of a multidimensional vector space. We utilize this hashcode as the index while inquiring. The order of this operation is O(1) (constant). This index refers to a bucket that contains data points that are spatially related. Only these modest fractions of data points must be used in the search. LSH splits the space in L different ways to eliminate false positives. It basically indicates that there are L hash tables in a vector space. This

is because employing distinct random partitionings prevents the randomly initialized partition vector from putting similar data into separate buckets. LSH retrieves all data points that are shared by all hash table buckets [4].



VA files:

Vector Approximation files are used for similarity search in high-dimensional spaces. The VA files overcome the Dimensionality Curse by using the filter-based approach of signature files rather than data partitioning approach of conventional index methods. Space rather than data is divided into cells and vectors are assigned approximations based on the cell which they lie in. Hence, the VA file is used as a simple filter, much as a signature file is a filter [5].

Singular Value Decomposition:

The factorization of a matrix A into the product of three matrices $A = UDV^T$, where the columns of U and V are orthonormal and the matrix D is diagonal with positive real entries, is known as singular value decomposition. The SVD is useful for a variety of activities. First, the data matrix A is often near to a low rank matrix, and it is useful to locate a low rank matrix that is a good approximation to the data matrix.

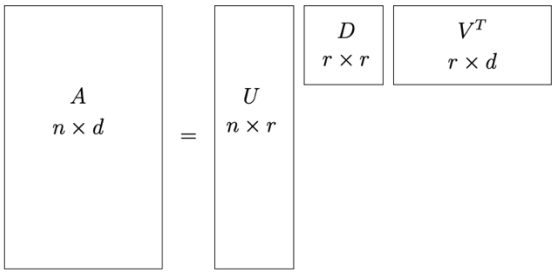


Figure 1.2: The SVD decomposition of an $n \times d$ matrix.

False Positive Rate (Fallout):

False positive rate (FPR) is a measure of accuracy for a test: be it a medical diagnostic test, a machine learning model, or something else. In technical terms, the false positive rate is defined as the probability of falsely rejecting the null hypothesis. The false positive rate is calculated as FP/FP+TN, where FP is the number of false positives and TN is the number of true negatives (FP+TN being the total number of negatives). It's the probability that a false alarm will be raised: that a positive result will be given when the true value is negative.

Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

False Negative Rate (Miss Rate):

Problem Specification

Task 1: Implement a program which, given a folder of images, one of the three feature models, and a user specified value of k, computes k latent semantics (if not already computed and stored), and given a second folder of images, associates **X** labels to the images in the second

folder using the classifier selected by the user: an SVM classifier, a decision-tree classifier, or a PPR based classifier, in this latent space. Also compute and print false positive and miss rates.

Task 2: Implement a program which, given a folder of images, one of the three feature models, and a user specified value of k, computes k latent semantics (if not already computed and stored), and given a second folder of images, associates **Y** labels to the images in the second folder using the classifier selected by the user: an SVM classifier, a decision-tree classifier, or a PPR based classifier, in this latent space. Also compute and print false positive and miss rates.

Task 3: Implement a program which, given a folder of images, one of the three feature models, and a user specified value of k, computes k latent semantics (if not already computed and stored), and given a second folder of images, associates **Z** labels to the images in the second folder using the classifier selected by the user: an SVM classifier, a decision-tree classifier, or a PPR based classifier, in this latent space. Also compute and print false positive and miss rates.

Task 4: Implement a Locality Sensitive Hashing (LSH) tool, which takes as input (a) the number of layers, L, (b) the number of hashes per layer, κ, and (c) a set of vectors (generated by other tasks) as input and creates an in-memory index structure containing the given set of vectors.

– Implement similar image search using this index structure:

∗ given a folder of images and one of the three feature models, the images are stored in an LSH data structure (the program also outputs the size of the index structure in bytes), and

∗ given image and t, the tool outputs the t most similar images; it also outputs · the numbers of buckets searched as well as the unique and overall number of images considered · false positive and miss rates.

Task 5: Implement a VA-file index tool and associated nearest neighbor search operations. The data structures and relevant algorithms are described in the following two papers.
Given (a) a parameter b denoting the number of bits per dimensions used for compressing the vector data and (b) a set of vectors (generated by other tasks) as input, the program and creates an in-memory index structure containing the indexes of the given set of vectors. The program also outputs the size of the index structure in bytes.

– Implement similar image search using this index structure:

∗ given a folder of images and one of the three feature models, the images are stored in a VA-file data structure (the program also outputs the size of the index structure in bytes), and

∗ given image and t, the tool outputs the t most similar images; it also outputs

· the numbers of buckets searched as well as the unique and overall number of images considered

· false positive and miss rates.

Task 6: Decision-tree-based relevance feedback: Implement a decision tree based relevance feedback system to improve nearest neighbor matches, which enables the user to label some of the results returned by the search task as relevant or irrelevant and then returns a new set of ranked results, either by revising the query or by re-ordering the existing results.

Task 7: SVM-classifier-based relevance feedback: Implement an SVM based relevance feedback system to improve nearest neighbor matches, which enables the user to label some of the results returned by the search task as relevant or irrelevant and then returns a new set of ranked results, either by revising the query or by re-ordering the existing results

Task 8: Query and feedback interface: Implement a query interface, which allows the user to provide a query, relevant query parameters (including how many results to be returned). Query results are presented to the user in decreasing order of matching.
The result interface should also allow the user to provide positive and/or negative feedback for the ranked results returned by the system.
User feedback is then taken into account (either by revising the query or by re-ordering the results as appropriate) and a new set of ranked results are returned.

Assumptions

The following assumptions are made during the implementation of this phase of the project.

1. All images in the folder are of the same size and shape 64 x 64.
2. The images in the folder are grayscale and have normalized values [0-1].
3. The libraries used to compute the distance give accurate results.
4. If two photos have the same distance/similarity score, the image with the lowest numerical image ID is picked as the more similar.
5. We have used SVD as Dimensionality Reduction method as no other method was specified.
6. For this implementation, we are given 12 types, 40 subjects and 10 images per subject. After Exploratory analysis, it was observed that some of the subjects were having 9 images. To address this issue, we have dropped the images with image_id = 10. Hence, we now have the final dataset comprising 4320 images (12 x 40 x 9).

Description of the proposed solution/implementation

Preprocessing / Helper Functions:

Due to class imbalance problems, a function has been defined which balances the dataset and each class has a number of instances equal to the minimum from all classes. There are a number of helper functions defined in the notebook. The most significant being the Feature Extraction function. This function either extracts Color Moments, Extended Local Binary Pattern or Histogram of Oriented Gradients. Three classifiers namely Support Vector Machine, Decision Tree, and Personalized Pagerank are used for classifying images. The code for it is mentioned before actual task implementation.

Task 1:

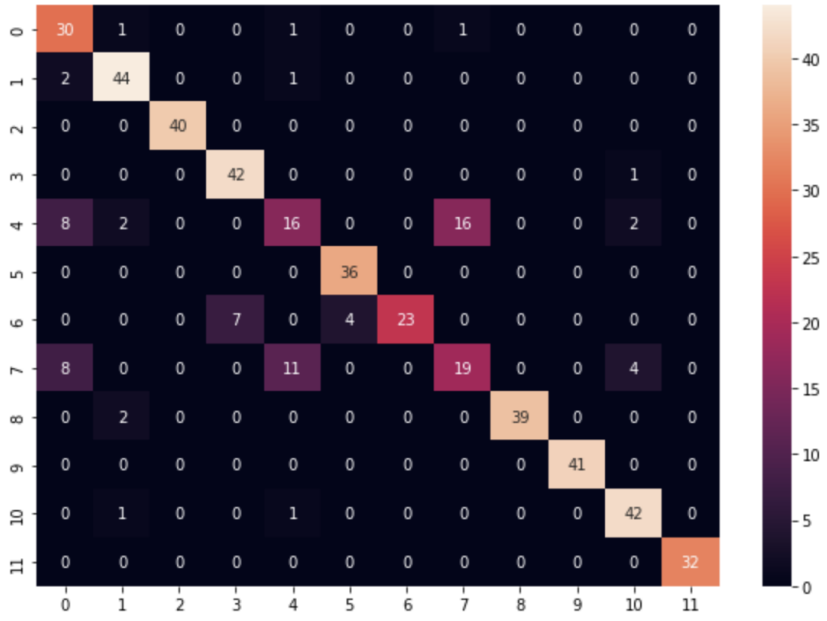
Input: *trainFolder, Feature model* (Color Moments, Extended Local Binary Pattern, Histogram of Oriented Gradients), *k* (Number of desired latent semantics), *testFolder*, *Classifier* (Decision Tree, Support Vector Machine, Personalized Pagerank)

Output: Each image of the *testFolder* is associated with a **X** label alongside the accuracy, false hits and misses.

Task 1 expects training data and testing data. The images are then used to extract features using CM, ELBP, and HOG and then dimensionality reduction (SVD) technique is used to reduce dims. Same modifications are also applied to test data. Now the reduced data along with labels is provided to classification models SVM, DT, and PPR. Then the accuracy is compared and results are analyzed.

Sample Output: We get one of the 12 classes (type) associated with each test image as output. The overall accuracy is mentioned in the table. Also, we get False Positive Rate (Fallout) and False Negative Rate (Miss Rate) for each class as showcased in the screenshot.

```
results1 = TASK1('train', 'HOG', 15, 'test', 'PPR')
```



```
Overall accuracy:  0.8469601677148847
Accuracy:  [0.95597484 0.98113208 1.         0.98322851 0.91194969 0.99161426
 0.9769392  0.91614256 0.99580713 1.         0.98113208 1.        ]

False positive rate (Fall out): [0.04054054 0.01395349 0.         0.01612903 0.03233256 0.00907029
 0.         0.03908046 0.         0.         0.01616628 0.        ]

False negative rate (Miss rate): [0.04402516 0.01886792 0.         0.01677149 0.08805031 0.00838574
 0.0230608  0.08385744 0.00419287 0.         0.01886792 0.        ]
```

We also tested a number of combinations and below are the results summarized:

Train	Test	k	Feature Model	Classifier	Accuracy (%)
2000	1000	25	CM	SVM	75
2000	1000	25	CM	DT	72
2000	1000	25	CM	PPR	65
2000	1000	25	ELBP	SVM	65
2000	1000	25	ELBP	DT	85
2000	1000	25	ELBP	PPR	88
2000	1000	25	HOG	SVM	88
2000	1000	25	HOG	DT	84
2000	1000	25	HOG	PPR	83

Task 2:

Input: *trainFolder, Feature model* (Color Moments, Extended Local Binary Pattern, Histogram of Oriented Gradients), *k* (Number of desired latent semantics), *testFolder*, *Classifier* (Decision Tree, Support Vector Machine, Personalized Pagerank)

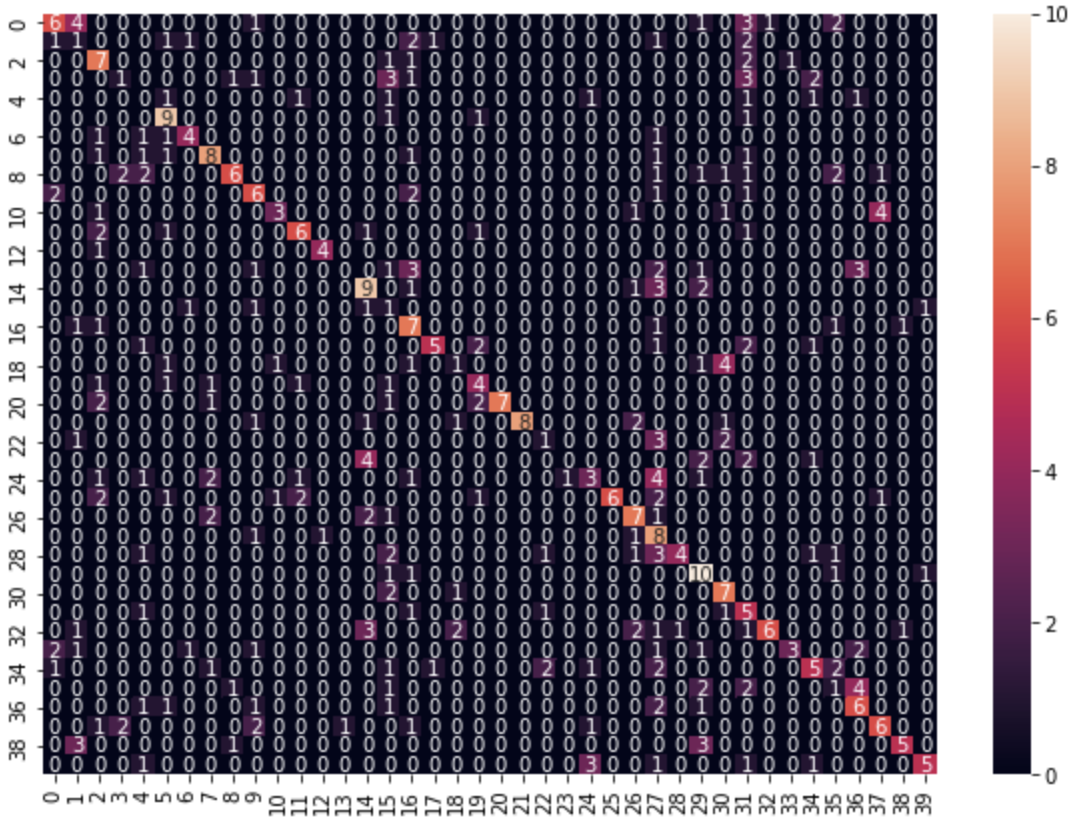
Output: Each image of the *testFolder* is associated with a **Y** label alongside the accuracy, false hits and misses.

Task 2 expects training data and testing data. The images are then used to extract features using CM, ELBP, and HOG and then dimensionality reduction (SVD) technique is used to reduce dims. Same modifications are also applied to test data. Now the reduced data along with labels is

provided to classification models SVM, DT, and PPR. Then the accuracy is compared and results are analyzed.

Sample Output: We get one of the 40 classes (subjects) associated with each test image as output. The overall accuracy is mentioned in the table. Also, we get False Positive Rate (Fallout) and False Negative Rate (Miss Rate) for each class as showcased in the screenshot.

```
results2 = TASK2('train', 'CM', 30, 'test', 'PPR')
```



```
Overall accuracy:  0.40041928721174
Accuracy:  [0.96226415 0.95807128 0.96016771 0.96855346 0.96226415 0.97484277
 0.98532495 0.97274633 0.9706499  0.96645702 0.98113208 0.9769392
 0.99580713 0.97274633 0.96016771 0.95387841 0.95597484 0.98113208
 0.97484277 0.97484277 0.98742138 0.98742138 0.97903564 0.97903564
 0.96226415 0.97903564 0.9706499  0.92662474 0.9769392  0.95807128
 0.97274633 0.94129979 0.97274633 0.97903564 0.96226415 0.96016771
 0.96436059 0.9706499  0.98113208 0.98113208]

False positive rate (Fall out): [0.0130719  0.0235546  0.03010753 0.00860215 0.02340426 0.01935484
 0.00639659 0.01511879 0.00652174 0.02150538 0.00428266 0.01075269
 0.00211864 0.00215054 0.02603037 0.03813559 0.0344086  0.00430108
 0.00854701 0.01495726 0.         0.         0.00851064 0.00213675
 0.01298701 0.         0.01724138 0.06866953 0.00215983 0.03455724
 0.02141328 0.05128205 0.00217865 0.00215054 0.01518438 0.0193133
 0.02155172 0.01295896 0.00430108 0.00430108]

False negative rate (Miss rate): [0.03773585 0.04192872 0.03983229 0.03144654 0.03773585 0.02515723
 0.01467505 0.02725367 0.0293501  0.03354298 0.01886792 0.0230608
 0.00419287 0.02725367 0.03983229 0.04612159 0.04402516 0.01886792
 0.02515723 0.02515723 0.01257862 0.01257862 0.02096436 0.02096436
 0.03773585 0.02096436 0.0293501  0.07337526 0.0230608  0.04192872
 0.02725367 0.05870021 0.02725367 0.02096436 0.03773585 0.03983229
 0.03563941 0.0293501  0.01886792 0.01886792]
```

We also tested a number of combinations and below are the results summarized:

Train	Test	k	Feature Model	Classifier	Accuracy (%)
2000	1000	25	CM	SVM	31
2000	1000	25	CM	DT	40
2000	1000	25	CM	PPR	61
2000	1000	25	ELBP	SVM	5
2000	1000	25	ELBP	DT	13
2000	1000	25	ELBP	PPR	17
2000	1000	25	HOG	SVM	64
2000	1000	25	HOG	DT	36
2000	1000	25	HOG	PPR	45

Task 3:

Input: *trainFolder, Feature model* (Color Moments, Extended Local Binary Pattern, Histogram of Oriented Gradients), *k* (Number of desired latent semantics), *testFolder*, *Classifier* (Decision Tree, Support Vector Machine, Personalized Pagerank)

Output: Each image of the *testFolder* is associated with a **Z** label alongside the accuracy, false hits and misses.

Task 3 expects training data and testing data. The images are then used to extract features using CM, ELBP, and HOG and then dimensionality reduction (SVD) technique is used to reduce dims. Same modifications are also applied to test data. Now the reduced data along with labels is provided to classification models SVM, DT, and PPR. Then the accuracy is compared and results are analyzed.

Sample Output: We get one of the 10 classes (sample ID) associated with each test image as output. The overall accuracy is mentioned in the table. Also, we get False Positive Rate (Fallout) and False Negative Rate (Miss Rate) for each class as showcased in the screenshot.

Train	Test	k	Feature Model	Classifier	Accuracy (%)
1000	500	25	CM	SVM	11
1000	500	25	CM	DT	32
1000	500	25	CM	PPR	21
1000	500	25	ELBP	SVM	10
1000	500	25	ELBP	DT	17
1000	500	25	ELBP	PPR	14
1000	500	25	HOG	SVM	13
1000	500	25	HOG	DT	23
1000	500	25	HOG	PPR	16

Task 4:

a) Input: *L* (Number of layers), *k* (Number of desired latent semantics), *trainFolder*, *Feature Model* (Color Moments, Extended Local Binary Pattern, Histogram of Oriented Gradients)

   Output: Creates an in-memory index structure containing the given set of vectors.

```
points_dict, w_length, config, train_folder, image_features,
reduced_data, reduced_dict, v_matrix = Task4a(10, 5, 'train', 'HOG')
```

LSH is a hashing approach which tries to maximize collisions to get similar documents. After balancing the dataset and reducing the dimensions, the reduced data is passed to *ImageLocalSensitiveHashing()* function along with *L* : number of layers, which returns index structures. This implementation is compatible to be used with HOG only as a feature model.

b) Input: *trainFolder*, *Feature Model* (Color Moments, Extended Local Binary Pattern, Histogram of Oriented Gradients), *imageID*, *t* (Number of most similar images)

Output: Returns *t* most similar images, the numbers of buckets searched as well as the unique and overall number of images considered and false positive and miss rates.

```
similar_images, query_image_id = Task4b(20, 'image-cc-1-1.png',
points_dict, w_length, config, train_folder, image_features,
reduced_data, reduced_dict, v_matrix)
```

```
Number of most similar images (t): 20
Query image name from the test folder: image-cc-1-1.png
Initializing LSH index with 10 Layers and 5 Hashes
Hash Key List ['40-48-41-50-48', '48-47-41-41-33', '43-41-37-34-54',
Getting bucket - 40-48-41-50-48
Getting bucket - 48-47-41-41-33
Getting bucket - 43-41-37-34-54
Getting bucket - 50-52-44-39-44
Getting bucket - 42-39-45-49-38
Getting bucket - 45-41-47-49-36
Getting bucket - 50-35-38-41-54
Getting bucket - 39-41-54-40-46
Getting bucket - 49-42-47-46-47
Getting bucket - 50-50-51-40-40
Getting bucket - 40-48-41-50
```

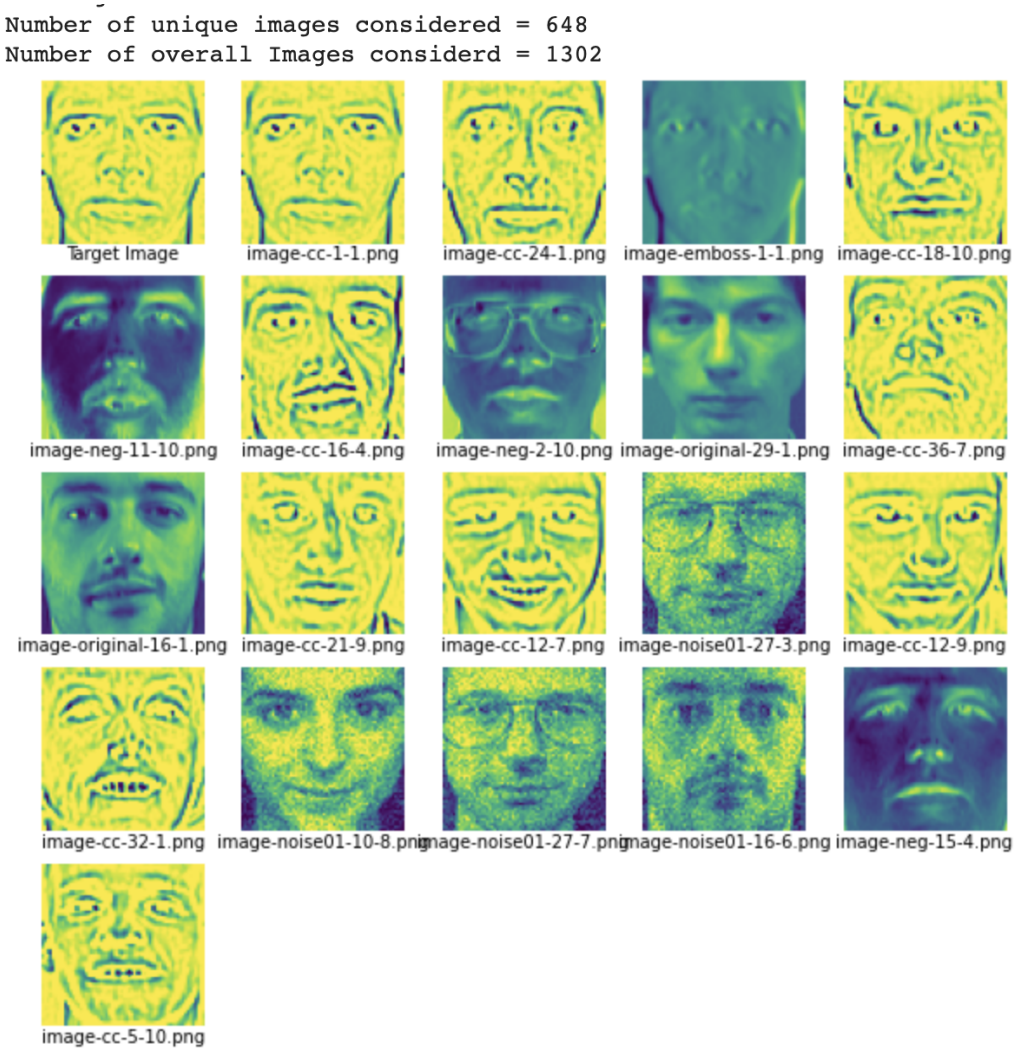
L / H	H = 5	H = 10	H = 15	H = 20
L = 5	60	60	65	65
L = 10	65	60	65	65
L = 15	65	65	65	65
L = 20	65	60	65	65

Accuracy Comparison (%)

L / H	H = 5	H = 10	H = 15	H = 20
L = 5	8	8	7	7
L = 10	7	8	7	7
L = 15	7	7	7	7
L = 20	7	8	7	7

Miss Rate Comparison (%)

Where L = Number of Layers;
H = Number of Hashes.



Sample Output: Images with ‘cc’ are correctly classified

Task 5:

Input: *b* (number of bits per dimensions used for compressing the vector data) , a set of vectors (generated by other tasks)

a) Input: *b* (number of bits per dimensions used for compressing the vector data) , *trainFolder*, *Feature Model* (Color Moments, Extended Local Binary Pattern, Histogram of Oriented Gradients)

Output: Creates an in-memory index structure containing the given set of vectors.

```
dict_aprroximation, size_of_dict_aprroximation, region_points,
partition_points = VA_files(int(input('Value of b: ')), U)
```

VA files is a filter based approach for similarity search. After balancing the dataset and reducing the dimensions, the reduced data is passed to *VA_files()* function along with *b* : number of bits, which returns index structures. This implementation is compatible to be used with CM, ELBP, andHOG only as a feature model.

b) Input: *trainFolder*, *Feature Model* (Color Moments, Extended Local Binary Pattern, Histogram of Oriented Gradients), *imageID*, *t* (Number of most similar images)

Output: Returns *t* most similar images, the numbers of buckets searched as well as the unique and overall number of images considered and false positive and miss rates.

```
query_image_id = input('Query image id: ')
similar_images, temp = Task5b("test/"+query_image_id, input('Feature
extraction model: '), 15)
```

```
Query image id: image-cc-1-1.png
Feature extraction model: HOG
The number of buckets searched  754
The number of overall images searched  765
The number of unique images searched  765
```

Feature Model	k	b	Accuracy (%)	Misses
CM	25	5	100	0

CM	25	14	100	0
CM	25	24	100	0
ELBP	25	5	67	5
ELBP	25	14	67	5
ELBP	25	24	67	5
HOG	25	5	33	10
HOG	25	14	33	10
HOG	25	24	40	9

```
['image-cc-1-1.png',
 'image-cc-24-1.png',
 'image-cc-18-10.png',
 'image-cc-16-4.png',
 'image-cc-21-9.png',
 'image-original-27-1.png',
 'image-cc-36-3.png',
 'image-cc-36-7.png',
 'image-original-16-1.png',
 'image-neg-18-8.png',
 'image-cc-19-6.png',
 'image-cc-39-3.png',
 'image-neg-11-10.png',
 'image-original-36-2.png',
 'image-cc-32-1.png']
```

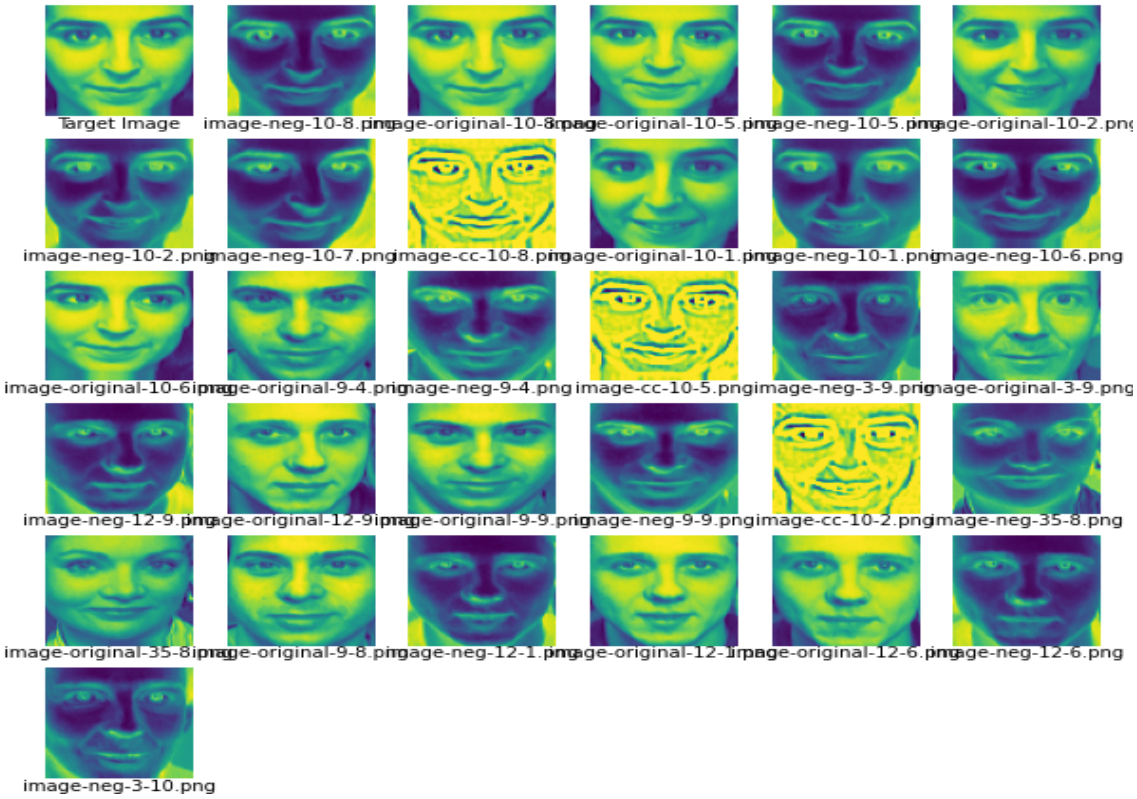
```
Accuracy for X:  0.6666666666666666
Misses for X:  5
Overall accuracy:  0.6666666666666666
Accuracy:  [0.66666667 0.66666667]

False positive rate (Fall out): [0.33333333        nan]

False negative rate (Miss rate): [0.33333333 0.33333333]
```

Task 6:

Input: a) Results from the search task which was implemented with LSH (Locality Sensitive Hashing, VA files).
b) Select from the search task results to label as relevant or irrelevant.

Output: Returns the new set of ranked results of the search task results based on the Decision tree based relevance feedback system.

Input to the model:search results output image

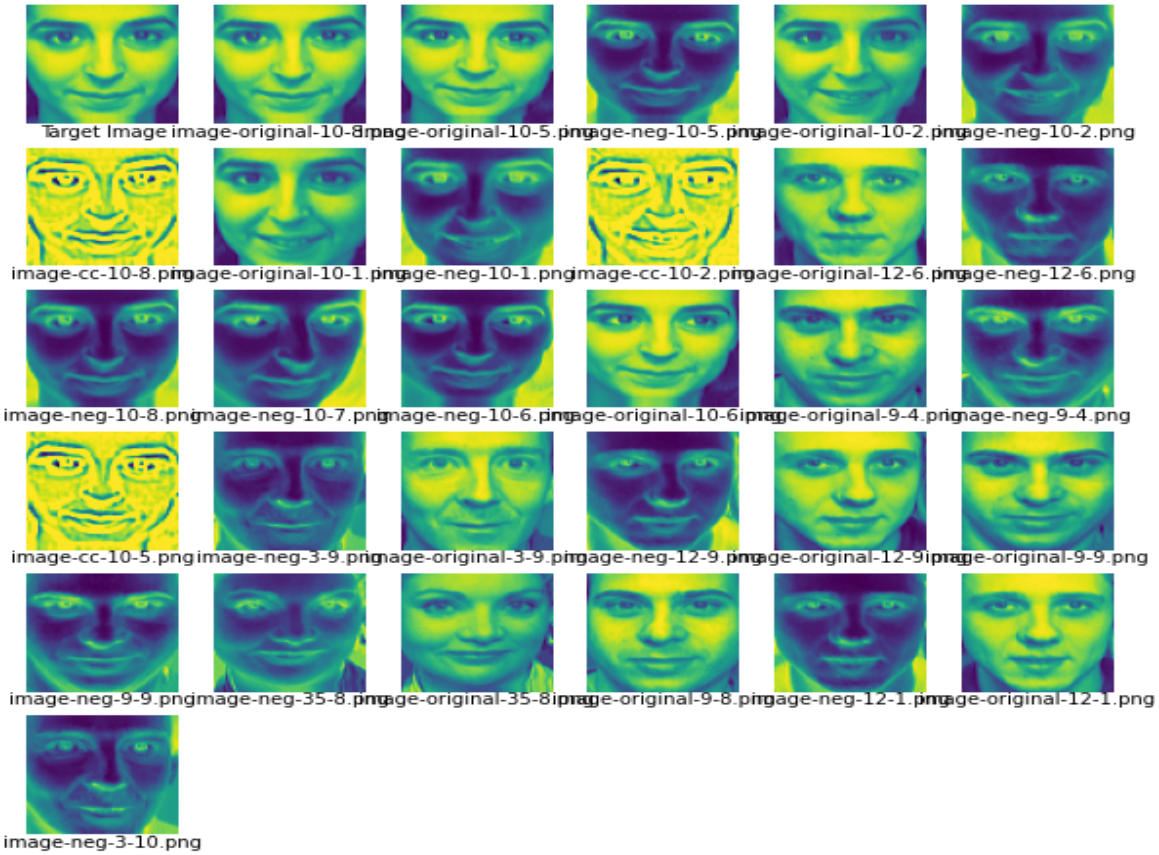


User feedback on search results:

```
Number of Relevant Images?: 5
Number of Irrelevant Images?: 5
1 image-neg-10-8.png
2 image-original-10-8.png
3 image-original-10-5.png
4 image-neg-10-5.png
5 image-original-10-2.png
6 image-neg-10-2.png
7 image-neg-10-7.png
8 image-cc-10-8.png
9 image-original-10-1.png
10 image-neg-10-1.png
11 image-neg-10-6.png
12 image-original-10-6.png
13 image-original-9-4.png
14 image-neg-9-4.png
15 image-cc-10-5.png
16 image-neg-3-9.png
17 image-original-3-9.png
18 image-neg-12-9.png
19 image-original-12-9.png
20 image-original-9-9.png
21 image-neg-9-9.png
22 image-cc-10-2.png
23 image-neg-35-8.png
24 image-original-35-8.png
25 image-original-9-8.png
26 image-neg-12-1.png
27 image-original-12-1.png
28 image-original-12-6.png
29 image-neg-12-6.png
30 image-neg-3-10.png
```

Output after reranking based on the Decision Tree feedback system:

```
image-original-10-8.png Relevant
image-original-10-5.png Relevant
image-neg-10-5.png Relevant
image-original-10-2.png Relevant
image-neg-10-2.png Relevant
image-cc-10-8.png Relevant
image-original-10-1.png Relevant
image-neg-10-1.png Relevant
image-cc-10-2.png Relevant
image-original-12-6.png Relevant
image-neg-12-6.png Relevant
image-neg-10-8.png Irrelavant
image-neg-10-7.png Irrelavant
image-neg-10-6.png Irrelavant
image-original-10-6.png Irrelavant
image-original-9-4.png Irrelavant
image-neg-9-4.png Irrelavant
image-cc-10-5.png Irrelavant
image-neg-3-9.png Irrelavant
image-original-3-9.png Irrelavant
image-neg-12-9.png Irrelavant
image-original-12-9.png Irrelavant
image-original-9-9.png Irrelavant
image-neg-9-9.png Irrelavant
image-neg-35-8.png Irrelavant
image-original-35-8.png Irrelavant
image-original-9-8.png Irrelavant
image-neg-12-1.png Irrelavant
image-original-12-1.png Irrelavant
image-neg-3-10.png Irrelavant
```



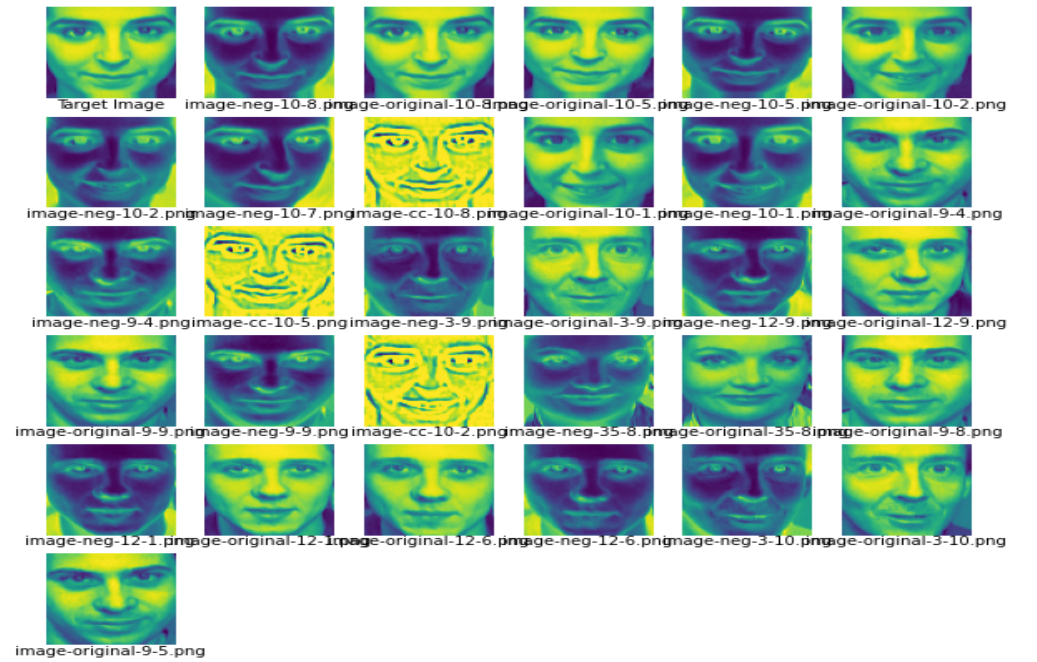
Task 7:

Input: a) Results from the search task which was implemented with LSH (Locality Sensitive hashing ,VA files).
b) Select from the search task results to label as relevant or irrelevant.

Output: Returns the new set of ranked results of the search task results based on the SVM classifier based relevance feedback system.

Input to the model:search results from task 4 and 5

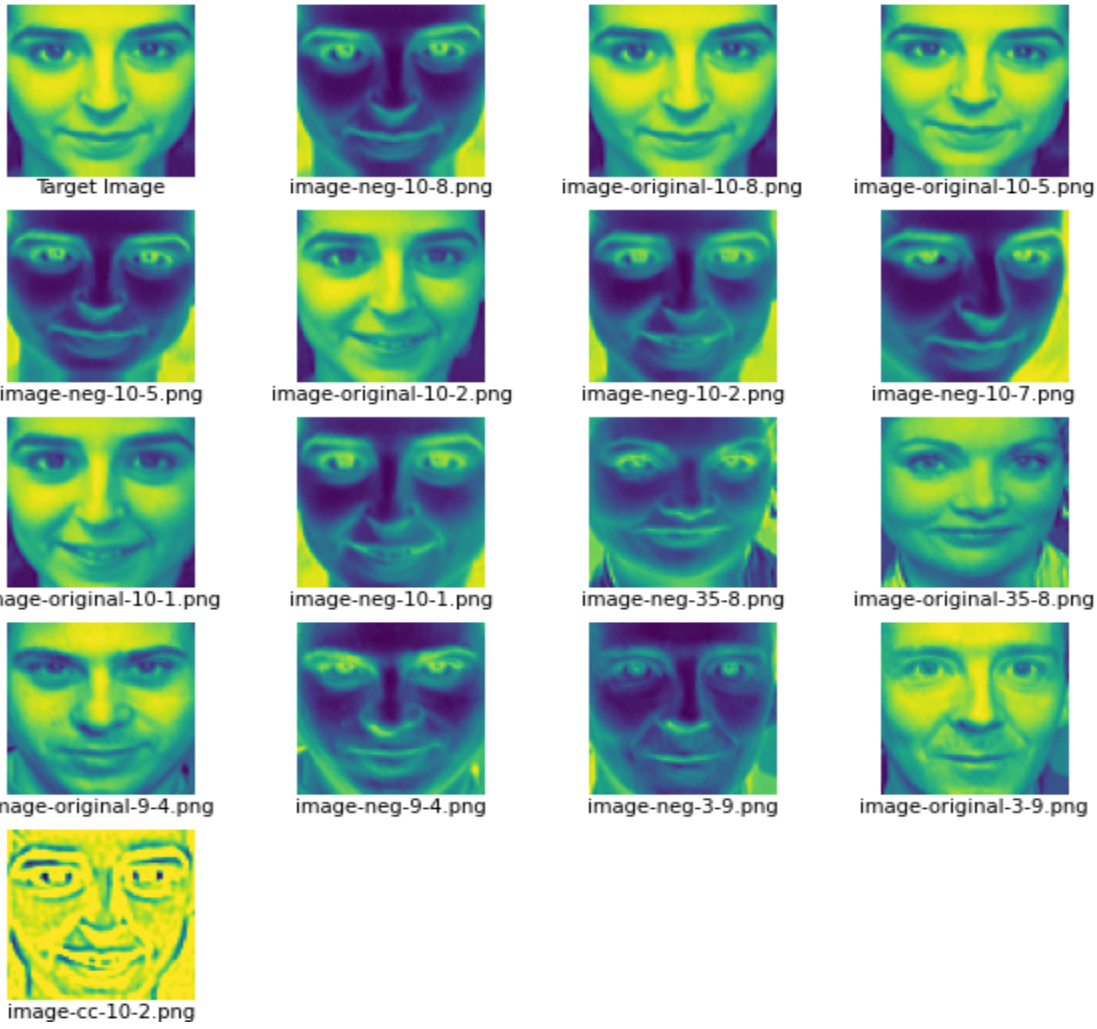
```
image-neg-10-8.png Relevant
image-original-10-8.png Relevant
image-original-10-5.png Relevant
image-neg-10-5.png Relevant
image-original-10-2.png Relevant
image-neg-10-2.png Relevant
image-neg-10-7.png Relevant
image-original-10-1.png Relevant
image-neg-10-1.png Relevant
image-neg-35-8.png Relevant
image-original-35-8.png Relevant
image-original-9-4.png Irrelavant
image-neg-9-4.png Irrelavant
image-neg-3-9.png Irrelavant
image-original-3-9.png Irrelavant
image-cc-10-2.png Irrelavant
```



User feedback on search results:

```
Relevant Image Number?:  2
Relevant Image Number?:  3
Relevant Image Number?:  5
Relevant Image Number?:  10
Relevant Image Number?:  1
Relevant Image Number?:  4
Relevant Image Number?:  20
Relevant Image Number?:  9
Relevant Image Number?:  21
Relevant Image Number?:  22
Irrelevant Image Number?:  11
Irrelevant Image Number?:  12
Irrelevant Image Number?:  14
Irrelevant Image Number?:  15
Irrelevant Image Number?:  20
```

Output after reranking based on the SVM based feedback system:



Task 8:

Input: a) query image and query parameters(SVM,Decision Tree) from the user.
b)select the relevant and irrelevant results from the decreasing ordered matching results result.
c)Take the feedback (-ve or +ve) based on relevant and irrelevant inputs given.

Output: This is used as the interface for task 4 and task 5.

Select the tool for indexing structure:

```
Select Indexing Structure
1.LSH
2.VA Files
2
```

Output from the search task:

```
Number of Relevant Images?: 10
Number of Irrelevant Images?: 5
1 image-neg-10-8.png
2 image-original-10-8.png
3 image-original-10-5.png
4 image-neg-10-5.png
5 image-original-10-2.png
6 image-neg-10-2.png
7 image-neg-10-7.png
8 image-cc-10-8.png
9 image-original-10-1.png
10 image-neg-10-1.png
11 image-original-9-4.png
12 image-neg-9-4.png
13 image-cc-10-5.png
14 image-neg-3-9.png
15 image-original-3-9.png
16 image-neg-12-9.png
17 image-original-12-9.png
18 image-original-9-9.png
19 image-neg-9-9.png
20 image-cc-10-2.png
21 image-neg-35-8.png
22 image-original-35-8.png
23 image-original-9-8.png
24 image-neg-12-1.png
25 image-original-12-1.png
26 image-original-12-6.png
27 image-neg-12-6.png
28 image-neg-3-10.png
29 image-original-3-10.png
30 image-original-9-5.png
```

Select the classification technique that you want to run for the search data:

```
Classifier
1. Decision Tree
2.SVM
 2
```

Input from the user as feedback based on the search results:

```
Relevant Image Number?:  2
Relevant Image Number?:  3
Relevant Image Number?:  5
Relevant Image Number?:  10
Relevant Image Number?:  1
Relevant Image Number?:  4
Relevant Image Number?:  20
Relevant Image Number?:  9
Relevant Image Number?:  21
Relevant Image Number?:  22
Irrelevant Image Number?:  11
Irrelevant Image Number?:  12
Irrelevant Image Number?:  14
Irrelevant Image Number?:  15
Irrelevant Image Number?:  20
```

System requirements /installation and execution instructions

The solution will work errorless with Python 3.6+.

System: Google colab (Tested only on this platform)

Requirements:
numpy==1.19.5
scipy==1.4.1
scikit-image==0.16.2
scikit-learn==0.22.2.post1
matplotlib==3.2.2
seaborn==0.11.2

Interface Specifications

Run the “CSE515 project phase3.ipynb” file into Google colab because there will be no dependency and environment issues and code will run smoothly.

Execution steps: -

1) Open the CSE515 project phase3.ipynb file.
2) Hit “**Run all**” from the “Runtime” tab.

3) After successful run, the cells above TASK 1 cell will import the libraries along with initializing helper functions.
4) After a successful run and providing required inputs, TASK 1 will output the required results and associate class X to all test images.
5) After a successful run and providing required inputs, TASK 2 will output the required results and associate class Y to all test images.
6) After a successful run and providing required inputs, TASK 3 will output the required results and associate class Z to all test images.
7) After a successful run and providing required inputs, TASK 4 will output the required results and classify test images based on the LSH method.
8) After a successful run and providing required inputs, TASK 5 will output the required results and classify test images based on the VA method.
9) After a successful run and providing required inputs, TASK 6 and TASK 7 will output nothing and will be used by TASK 8.
10) After a successful run and providing required inputs, TASK 8 will opt for parameters and produce results as expected. This might ask for user input at different steps.

Conclusions

It can be concluded that various feature extraction models lead to various results. Different feature extraction techniques capture different aspects of images and hence lead to different outcomes. Also, different classification methods are used in the work, which again work differently for different inputs. Hence, it can be concluded that Personalized Pagerank along with Histogram of Oriented Gradients gave higher accuracy with lower miss rate. Also, Hashing methods like Locality Sensitive Hashing and Vector Approximation produced better results with high accuracy and low miss rates.

Bibliography

1. Aggarwal, C.C., Hinneburg, A. and Keim, D.A., 2001, January. On the surprising behavior of distance metrics in high dimensional space. In *International conference on database theory* (pp. 420-434). Springer, Berlin, Heidelberg.
2. Page, L., Brin, S., Motwani, R. and Winograd, T., 1999. The PageRank citation ranking: Bringing order to the web. Stanford InfoLab.
3. Lofgren, P., Banerjee, S. and Goel, A., 2016, February. Personalized pagerank estimation and search: A bidirectional approach. In Proceedings of the Ninth ACM International Conference on Web Search and Data Mining (pp. 163-172).
4. Loïc Paulevé, Hervé Jégou, Laurent Amsaleg. Locality sensitive hashing: a comparison of hash function types and querying mechanisms. Pattern Recognition Letters, Elsevier, 2010, 31 (11), pp.1348-1358.
5. A Simple Vector-Approximation File for Similarity Search in High-Dimensional Vector Spaces (1997).

Appendix

Specific roles of the group members

Each team member was responsible for implementation of the assigned tasks of this phase of the project on their own. The collaboration was confined to the overall solution design and idea sharing.