CSE515 - Multimedia and Web Databases - Fall 2023

Project Report - Phase 2

Kaushik Ravishankar, Madhura Bhalchandra Wani, Mohankrishna Chandrashekar

Niraj Sonje, Pavan Rathnakar Shetty, Pranav Borikar

ABSTRACT

In the second phase of this project, we delve into latent spaces, dimensionality curse and reduction, graph analysis and more. We first continue from the first phase of the project where we left off, at feature extraction from the labeled dataset to get the five specified feature spaces. In addition, we also extract the feature space generated by the ResNet50 model when passed through the full model, and apply a softmax activation function. Then in the first two tasks, using these feature spaces, given images or labels, we find similar images or labels as needed, using appropriate similarity measures.

Dimensionality reduction and latent semantics extraction are explored in Tasks 3 to 6, to obtain hidden patterns from the feature spaces. Namely, we use Singular Value Decomposition (SVD), Non-Negative Matrix Factorization (NNMF), Latent Dirichlet Allocation (LDA) and k-means clustering algorithms. Then, in Tasks 7 through 10, these latent semantics are used for performing label matching and finding image similarity under various scenarios and latent spaces. The final task involves similarity graph construction and application of personalized PageRank algorithm to find the most important images in the chosen feature or latent space in relation to a specific label.

The goal of this project is to develop a flexible system for image retrieval and analysis. It will offer a wide range of features, such as feature extraction, latent semantics analysis, and tailored content suggestion, allowing users to explore and engage with image data in a variety of ways.

Keywords

RESNET50, Caltech101, CM, HOG, ResNet-AvgPool-1024, ResNet-Layer3-1024, ResNet-FC-1000, k-images, query label-l, SVD, NNMF, LDA, k-means, latent semantics, CP-decomposition, weights, similarity matrix, dimensionality reduction, similarity graph

INTRODUCTION

The field of multimedia and web databases plays a pivotal role in the organization, analysis, and retrieval of vast collections of multimedia data, including images, videos, and web content. In the age of big data, it is vital for a variety of applications, from content recommendation to image search and retrieval, to be able to efficiently manage and extract meaningful information from multimedia sources. In this project, we embark on a comprehensive exploration of multimedia and web databases by addressing a series of tasks that encompass image features, vector models, dimensionality reduction, and graph analysis.

Key Terminology:

- *Features* - Unique attributes or traits derived from data, frequently employed for tasks such as analysis, identification, and categorization
- *Feature descriptors* - Representations of features in numerical, vectorized formats
- *Caltech101* - A dataset of 101 diverse categories of images, which was popularly used in object recognition benchmarks
- *ResNet50* - A residual neural network architecture which has 50 layers, exceptionally performant in image classification tasks, trained on the ImageNet 1K dataset
- *Distance/similarity measure* - Quantifier of the dissimilarity or similarity between two data points, helping to assess how close or different they are in a given feature space, commonly used for clustering, classification, and retrieval tasks in multimedia analysis. Examples include Euclidean distance, cosine similarity, Pearson similarity, Kullback-Leibler divergence.
- *Dimensionality Curse* - Trend of increase in complexity with increase in number of dimensions, which can make the computations more complex and reduce the effectiveness of data analysis.
- *Latent Semantics*: Hidden, meaningful patterns or features that are present in the data.
- *Singular Value Decomposition (SVD)*: Factorization of a matrix using eigenvalue decomposition techniques.
- *Non-negative Matrix Factorization (NNMF)*: Approximate factorization of a matrix using iterative trial-and-error methods.
- *Latent Dirichlet Allocation (LDA)*: Probabilistic approach to topic modeling based on Bayesian networks and Dirichlet distributions.
- *K-Means Clustering*: Unsupervised clustering algorithm that finds unrelated groups of related data.
- *Candecomp/Parafac (CP) Decomposition*: Factorization of a multi-dimensional tensor into a set of factor matrices and a core tensor. It can reveal complex latent patterns and relationships within the data.

- *Graph Analysis*: In this project, graph analysis entails constructing and analyzing graphs where nodes represent images, and edges represent similarities between images.
- *Personalized PageRank:* Variant of Google's original PageRank algorithm used to rank web pages relevant to a certain query.

Goal Description:

This project encompasses a comprehensive exploration of image data analysis. It includes a series of tasks aimed at extracting meaningful information from image data, reducing dimensionality, and conducting graph-based analysis.

The main objectives of this project are as follows:

Task 0: Feature Extraction and Image Retrieval

- Task 0 picks up from Phase 1 of the project, where we perform feature extraction using various feature models such as color moments, histogram oriented gradients and outputs of ResNet50 layers, and using those features to identify and visualize similar images for a given query image and feature model.

Task 1: Label-Based Image Retrieval

- The objective of Task 1 is to retrieve and visualize the 'k' most relevant images associated with a user-specified label within a feature space of their choice. This task enables users to quickly locate and explore images that share a common label, facilitating efficient image retrieval and analysis based on semantic information.

Task 2: Label Prediction

- In Task 2a, the objective is to predict and list the k most likely matching labels for a given query image within a chosen feature space.
- Task 2b focuses on predicting and listing the k most probable matching labels using the RESNET50 neural network model.

Task 3 (LS1): Latent Semantic Extraction (Feature Space)

- Task 3 consists of using dimensionality reduction techniques like SVD, NNMF, LDA, or k-means to extract the top-k latent semantics within a chosen feature space.

Task 4 (LS2): Latent Semantic Extraction (CP Decomposition)

- In this task, a three-modal tensor (image-feature-label) is used to examine the extraction of latent semantics using CP decomposition. The resulting latent semantics are presented as lists of label-weight pairs.

Task 5 (LS3): Label-Label Similarity Analysis

- Task 5 focuses on creating a label-label similarity matrix, reducing its dimensionality, and storing the resulting latent semantics in a file.

Task 6 (LS4): Image-Image Similarity Analysis

- This task involves creating an image-image similarity matrix, reducing its dimensionality, and storing the resulting latent semantics in a file.

Task 7: Image Retrieval from Latent Space

- This task involves identifying and visualizing the k most similar images to a given query image within a selected latent space.

Task 8: Label Prediction from Latent Space

- Task 8 centers on predicting and listing the k most probable matching labels for a given query image under a chosen latent space.

Task 9: Label Prediction Using Latent Semantics

- In this task, the objective is to predict and list the k most likely matching labels within a selected latent space for a given label.

Task 10: Image Retrieval Using Latent Semantics

- This task focuses on identifying and listing the k most relevant images for a given label within a chosen latent space.

Task 11: Similarity Graph Construction and Personalized PageRank

- The final task involves creating a similarity graph based on image similarities and utilizing personalized PageRank to identify the most significant images relative to a given label.

Assumptions:

We have made the following assumptions to complete these tasks:

1. The Caltech 101 dataset, containing labeled images, is available for feature extraction and analysis.
2. Use of pre-trained models, libraries, and tools for feature extraction, dimensionality reduction, and graph analysis is permitted to a certain extent.
3. Properly labeled images within the dataset facilitate label-based tasks.
4. Sufficient computational resources are accessible to handle dimensionality reduction and graph analysis.
5. For the purpose of ensuring effective task execution, intermediate results can be saved and loaded.

PROPOSED SOLUTION/IMPLEMENTATION

Environment, dataset and tools:

The environment uses Jupyter Notebooks, and preferably utilizes a GPU for faster processing. The Caltech101 dataset and ResNet50 model can both be downloaded using modules from TorchVision. All python libraries can be downloaded and installed using the requirements.txt file in the source code repository.

TASK 0

0A)

Specification:

This task requires using the pre-trained RESNET50 neural network model to map even numbered (labeled) images in the Caltech101 dataset into 5 different feature spaces: Color moments, Histograms of oriented gradients, ResNet-AvgPool-1024, ResNet-Layer3-1024, and ResNet-FC-1000. The resulting data vectors should be stored in a database, along with the imageIDs and original image labels.

Description:

The goal of this task is to extract features from the Caltech101 data set using the RESNET50 neural network model. These features can then be used for various machine learning tasks, such as image classification and object detection.

Design:

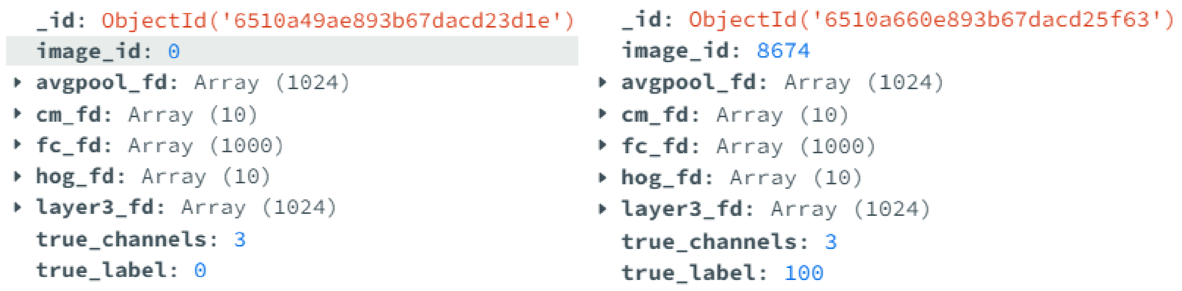
1. The image is resized to 300 x 100 and then partitioned into a 10x10 grid, slicing the original image.
2. **Color Moments:** Mean, standard deviation, and skewness of each of the 100 smaller grids are calculated for each color channel and these are then combined to form a feature descriptor of dimensional shape (10,10,3,3), i.e. 900 dimensions
3. **Histogram of oriented gradients (HOG):** The image is converted to grayscale using the OpenCV library and gradient change in horizontal and vertical directions are calculated using the Sobel operator, using the first-order central difference option of the function, using the kernel masks $dx = [-1, 0, 1]$ and $dy = [-1, 0, 1]^T$ for convolving with the image grid to get the gradients. The gradients are binned by their direction into 9 bins of equal ranges and weighted by their magnitude to form a histogram. These are then combined to form a feature descriptor of dimensional shape (10,10,9), i.e. 900 dimensions

4. **ResNet50 - avgpool, layer3 and fc layers:** The pre-trained ResNet50 model is loaded with its default weights using TorchVision. To the avgpool, layer3 and fc layers of the neural network, hooks that capture the output of the layer are attached.
    a. The feature descriptor of size 2048 obtained from the avgpool layer is shrunk to size 1024 by averaging consecutive pairs of elements.
    b. The feature descriptor of size 1024x14x14 obtained from the layer3 layer is shrunk to size 1024 by averaging along the 14x14 grids
    c. The feature descriptor of size 1000 is obtained from the fc layer
5. Features are extracted from the entire dataset and stored in a local instance of a MongoDB database. NoSQL is preferred to relational databases due to their inefficiency in storing large binary blobs and complexity when handling inconsistent data structures. The PyMongo module provides the database driver functionality between Python and MongoDB

Improvements done:

1. Skewness: during the first phase, the skewness computed was giving NaN values. For this phase, the NaN values were handled by checking whether the value in the cube root was negative or not, if found negative, the absolute value was computed and then the cube root was computed.

Output:



```
  _id: ObjectId('6510a49ae893b67dacd23d1e')
  image_id: 0
▸ avgpool_fd: Array (1024)
▸ cm_fd: Array (10)
▸ fc_fd: Array (1000)
▸ hog_fd: Array (10)
▸ layer3_fd: Array (1024)
  true_channels: 3
  true_label: 0
```

```
  _id: ObjectId('6510a660e893b67dacd25f63')
  image_id: 8674
▸ avgpool_fd: Array (1024)
▸ cm_fd: Array (10)
▸ fc_fd: Array (1000)
▸ hog_fd: Array (10)
▸ layer3_fd: Array (1024)
  true_channels: 3
  true_label: 100
```

The output above is obtained from MongoDB Compass, in this:

- **_id:** it is the default id value assigned by MongoDB and is unique for each document
- **image_id:** this represents the image id of the image in the caltech101 dataset
- **avgpool_fd:** represents the features obtained by the avgpool layer of resnet50
- **cm_fd:** represents the features obtained by the color moments
- **fc_fd:** represents the features obtained by the fully connected layer of resnet50
- **hog_fd:** represents the features obtained by histogram of oriented gradients
- **layer3_fd:** represents the features obtained by layer 3 of resnet50
- **true_channels:** represents the number of channels of that image, 3 for RGB and 1 for grayscale
- **true_label:** represents the label of the image according to caltech101 dataset

0B)

Specification:

In this task, various similarity measures are defined, tested and finalized to determine the top k similar images. The image ID, k-value and the feature model is taken as input and k-similar images of the image ID are displayed with their similarity scores

Description:

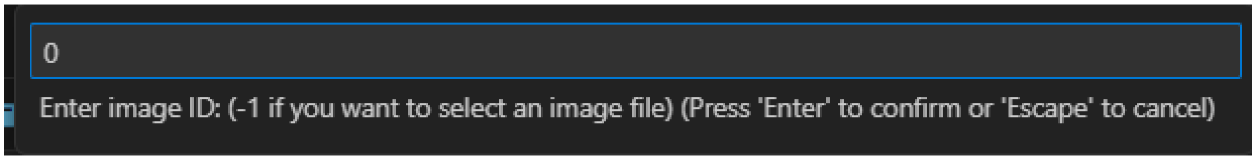
This program works by first extracting the feature vector from the input image using the selected feature space. It then loads all of the feature vectors from the database and calculates the similarity between the input image's feature vector and each of the other feature vectors. The k images with the highest similarity scores are then returned as output.

Design:

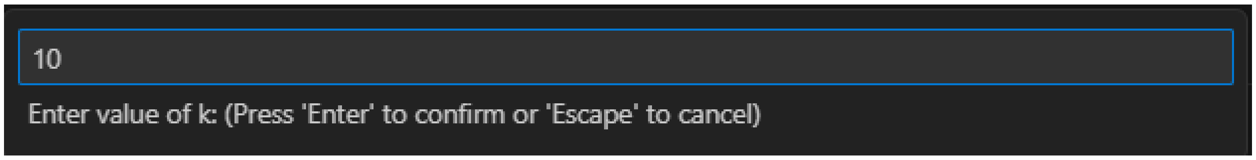
The similarity measures are defined to calculate the similarity and display the top k-similar images. Euclidean distance measure is used to calculate the similarity for color moments, cosine similarity for histogram of gradients and finally pearson similarity for avgpool layer, FC layer and layer 3 of resnet50.

By definition cosine similarity is useful to compare angles and directional differences, and a HOG is built on gradient orientations and hence it is used for determining similarity involving HOG.

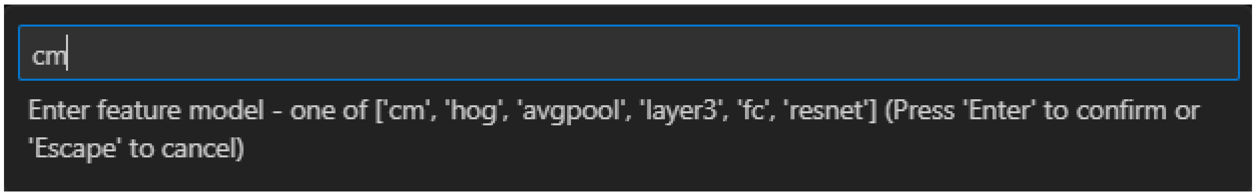
Input:



```
0
Enter image ID: (-1 if you want to select an image file) (Press 'Enter' to confirm or 'Escape' to cancel)
```



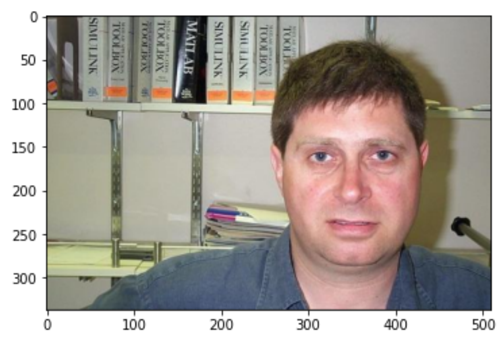
```
10
Enter value of k: (Press 'Enter' to confirm or 'Escape' to cancel)
```



```
cm
Enter feature model - one of ['cm', 'hog', 'avgpool', 'layer3', 'fc', 'resnet'] (Press 'Enter' to confirm or 'Escape' to cancel)
```

Output:

Image index=0:



Showing 10 similar images for image ID 0, using euclidean_distance_measure for cm_fd feature descriptor…

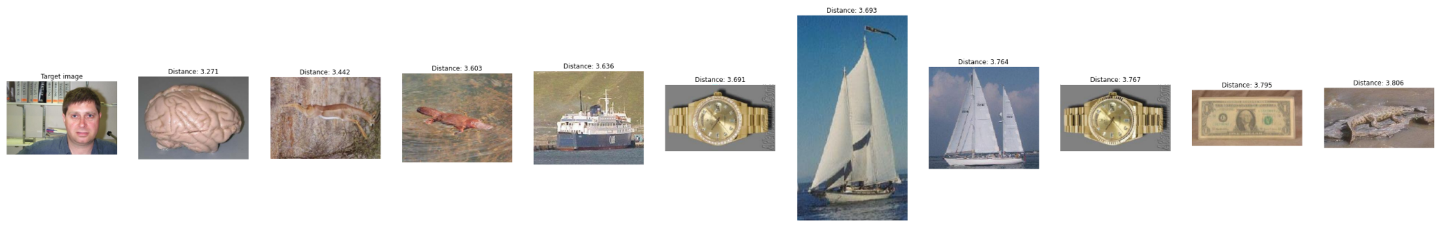
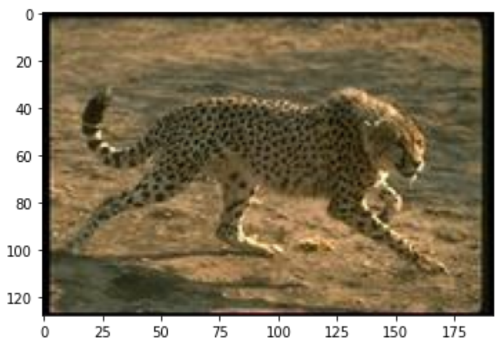


Image index=880



Showing 10 similar images for image ID 880, using pearson_distance_measure for avgpool_fd feature descriptor…

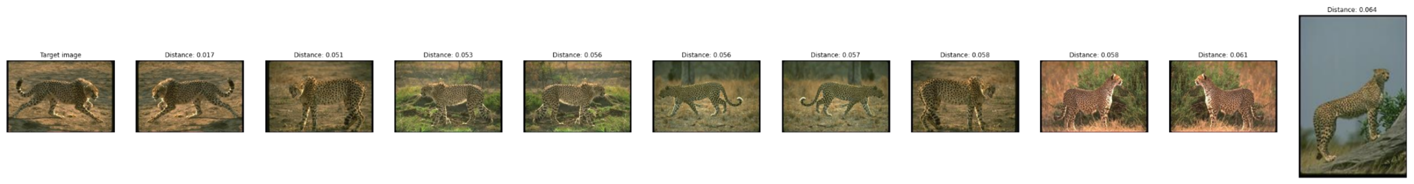
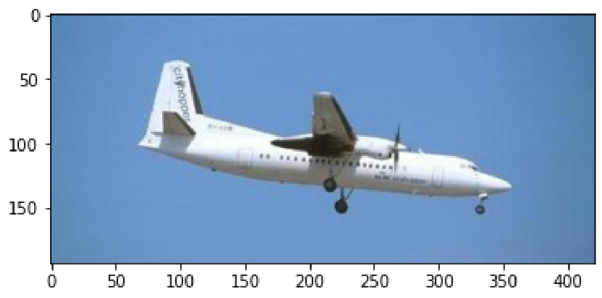


Image index=2500



Showing 10 similar images for image ID 2500, using pearson_distance_measure for layer3_fd feature descriptor...

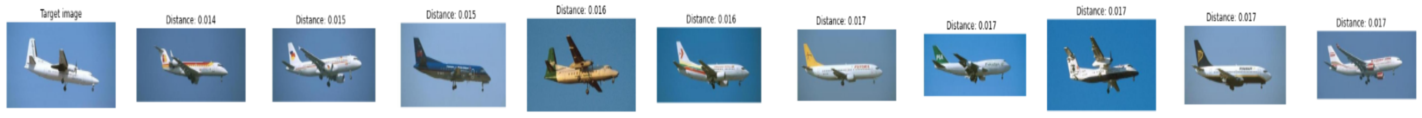
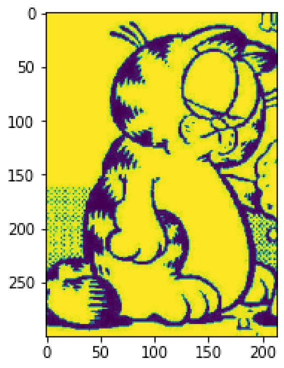


Image index=5122



Showing 10 similar images for image ID 5122, using cosine_distance_measure for hog_fd feature descriptor…

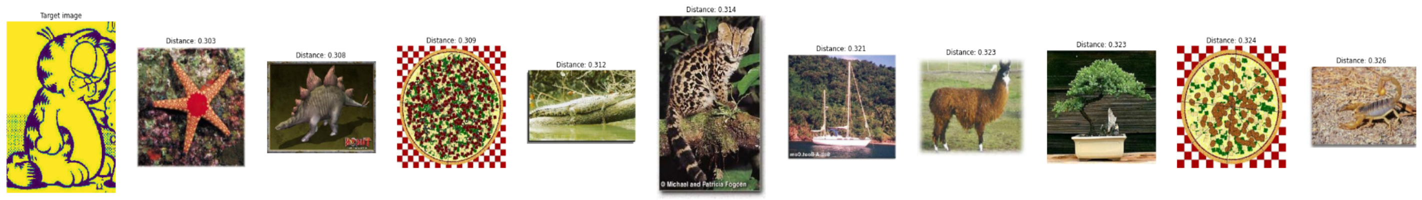
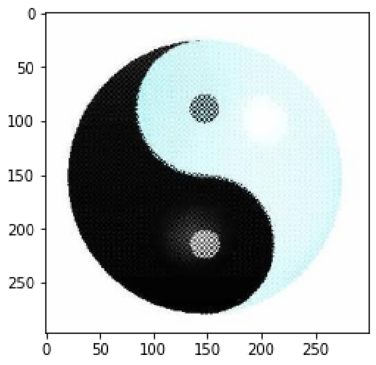
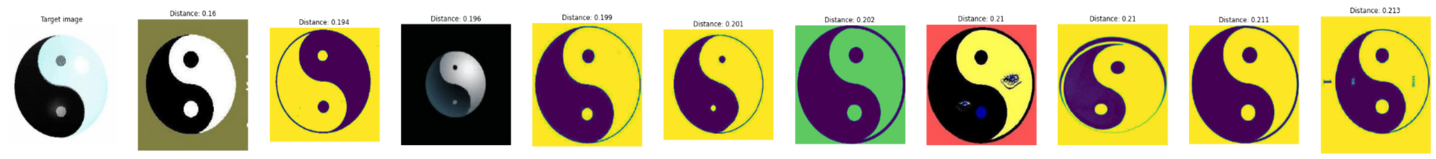


Image index=8676



Showing 10 similar images for image ID 8676, using pearson_distance_measure for fc_fd feature descriptor…



TASK 1

Specification:

The user inputs an image query label (0 to 100), the value K, one of the feature models (Colour Moments, HOG, Avgpool, Layer3 and FC). The function uses the specified feature model to extract feature vectors from the images in the database, and then uses the specified distance measure to calculate the distance between each image's feature vector and the target label's feature vector.
It will output k similar images based on the label and feature models given by the user.

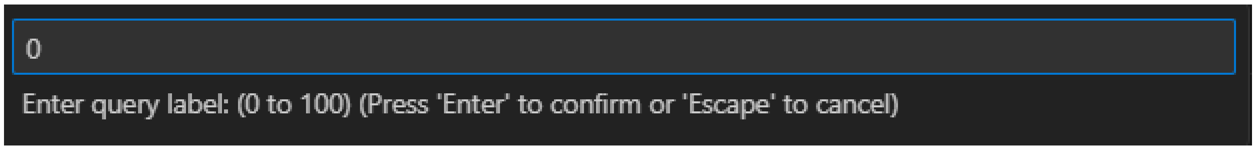
Description:

The function works by first calculating the representative feature vector for the target label. This is done by averaging the feature vectors of all images in the database that have the target label. Once the representative feature vector for the target label has been calculated, the function then calculates the distance between the representative feature vector and the feature vector of each image in the database. The function then returns the top k images with the smallest distances to the target label.

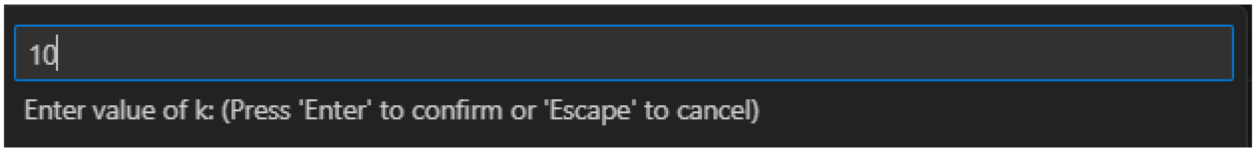
Design:

1. The representative feature vector in this case is calculated by computing the mean of the feature vector specified by the user of all the images which have the label, l, entered by the user. The label mean representative is calculated by adding up all the representatives under that label and dividing it by the total number of feature vectors present.
   label_mean_vector = [sum(col) / len(col) for col in zip(*label_fds)]
2. To find the k-most similar images, the mean feature representative is compared with the features of all the images and the k-most closest or similar feature spaces are obtained and the images corresponding to that are displayed.

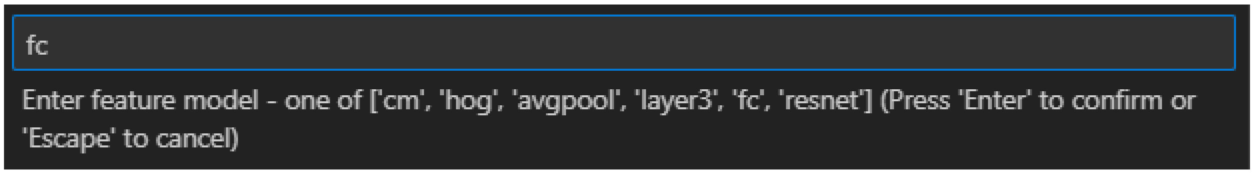
Input:



```
0
Enter query label: (0 to 100) (Press 'Enter' to confirm or 'Escape' to cancel)
```



```
10
Enter value of k: (Press 'Enter' to confirm or 'Escape' to cancel)
```

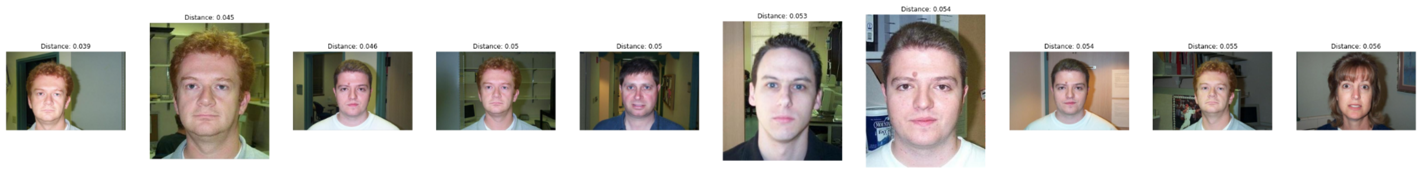


```
fc
Enter feature model - one of ['cm', 'hog', 'avgpool', 'layer3', 'fc', 'resnet'] (Press 'Enter' to confirm or 'Escape' to cancel)
```

Output:

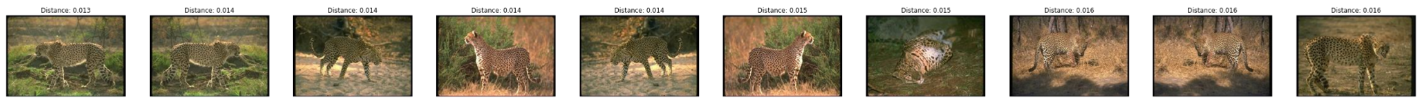
label=0

Showing 10 similar images for label 0, using pearson_distance_measure for fc_fd feature descriptor...



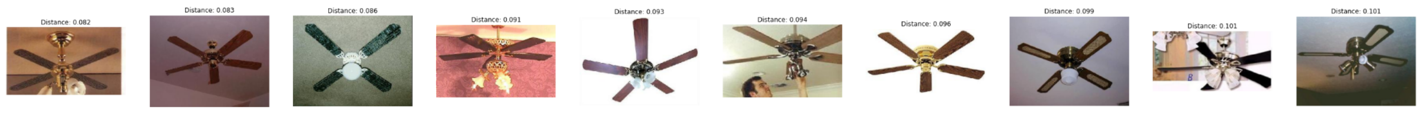
label=2

Showing 10 similar images for label 2, using pearson_distance_measure for layer3_fd feature descriptor...



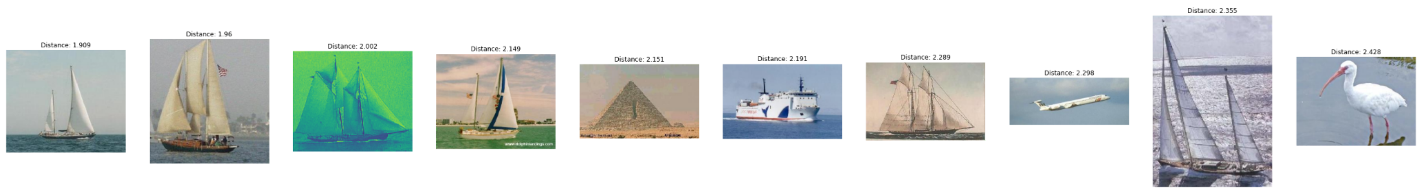
label=20

Showing 10 similar images for label 20, using pearson_distance_measure for avgpool_fd feature descriptor...



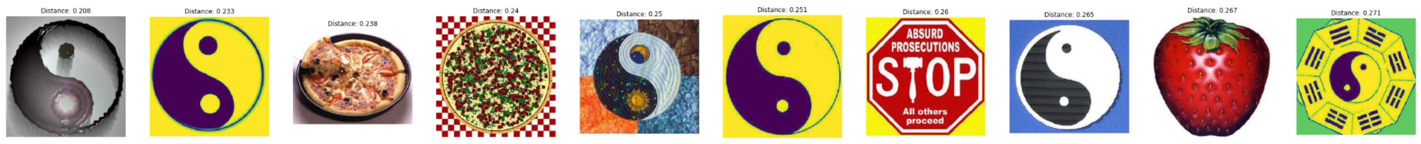
label=55

Showing 10 similar images for label 55, using euclidean_distance_measure for cm_fd feature descriptor...



label=100

Showing 10 similar images for label 100, using cosine_distance_measure for hog_fd feature descriptor...



TASK 2

2A)

Specification:

The user is required to input an Image ID or select an image file, the value K, one of the feature models (Colour Moments, HOG, Avgpool, Layer3 and FC). It will display k most similar labels according to the Image ID or image file and the selected feature space along with the scores.

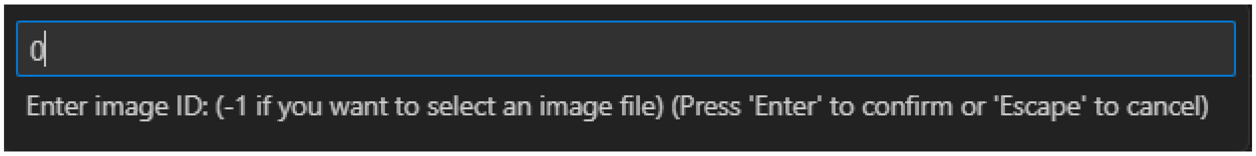
Description:

This program works by first extracting the feature vector from the query image using the selected feature space. It then calculates the similarity between the query image's feature vector and the feature vectors of all the images in the database. The k labels with the highest similarity scores are then returned as output.

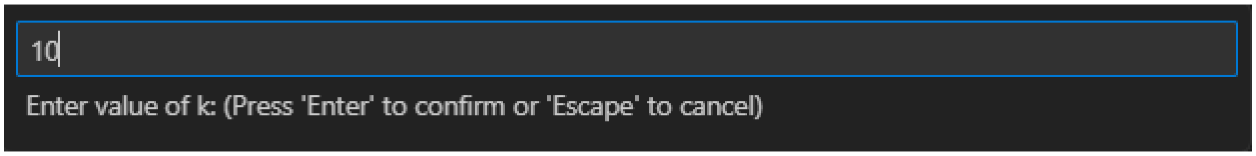
Design:

1. The image ID entered is searched for in the database, if not found the feature descriptors are calculated. If the input is an image file, all the values except the true_label value are determined.
2. Two dictionaries min_dists and label_dict are initialized in which the 'key' is the image ID and the 'value' is the similarity distance and label for min_dists and label_dict respectively.
3. Using the feature vectors and based on the type of feature space, the first k-most similar images with unique (non-repeated) labels are determined by simultaneously checking for the label value of the similar image.
4. The similarity is determined for all the images and the ones with even better similarity scores replace the ones with the worst similarity scores in both the dictionaries, simultaneously checking for non-repeated labels.
5. The min_dists dictionary is finally sorted in ascending order and the output is displayed with the label and its corresponding similarity value,
6. During this process the images with the same label as the input image are not compared.

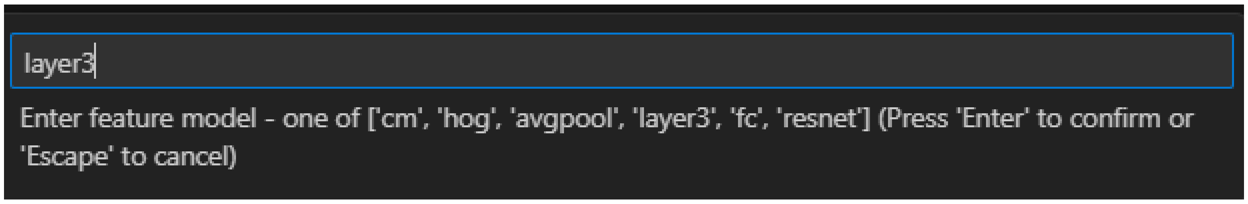
Input:



```
0
Enter image ID: (-1 if you want to select an image file) (Press 'Enter' to confirm or 'Escape' to cancel)
```



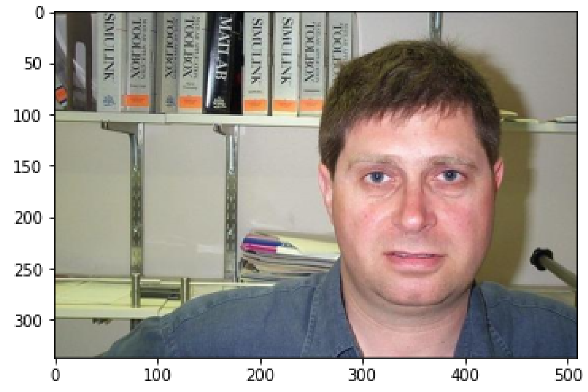
```
10
Enter value of k: (Press 'Enter' to confirm or 'Escape' to cancel)
```



```
layer3
Enter feature model - one of ['cm', 'hog', 'avgpool', 'layer3', 'fc', 'resnet'] (Press 'Enter' to confirm or 'Escape' to cancel)
```

Output:

Image index=0:



Showing 10 similar labels for image ID 0, using pearson_distance_measure for layer3_fd feature descriptor…

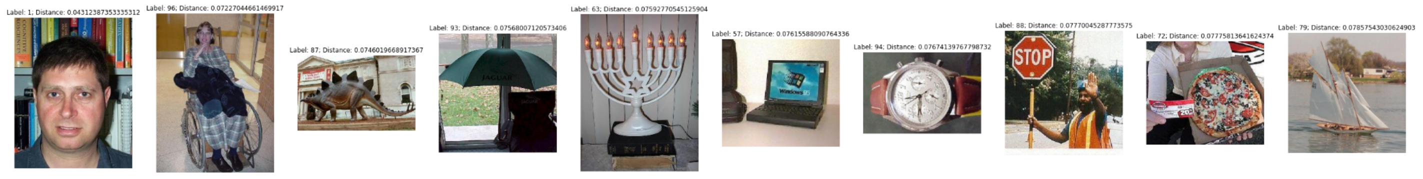
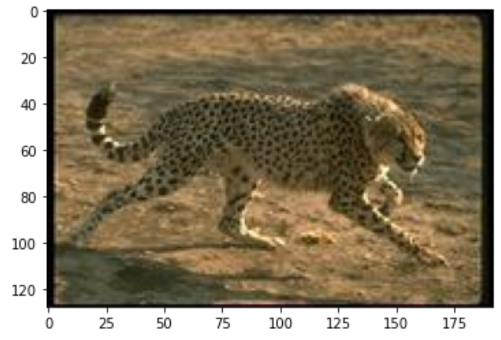

Label: 1; Distance: 0.04312387353335312
Label: 96; Distance: 0.07227044661469917
Label: 87; Distance: 0.0746019668917367
Label: 93; Distance: 0.07568007120573406
Label: 63; Distance: 0.07592770545125904
Label: 57; Distance: 0.07615588090764336
Label: 94; Distance: 0.07674139767798732
Label: 88; Distance: 0.07770045287773575
Label: 72; Distance: 0.07775813641624374
Label: 79; Distance: 0.07857543030624903

Image index=880:



Showing 10 similar labels for image ID 880, using pearson_distance_measure for fc_fd feature descriptor…

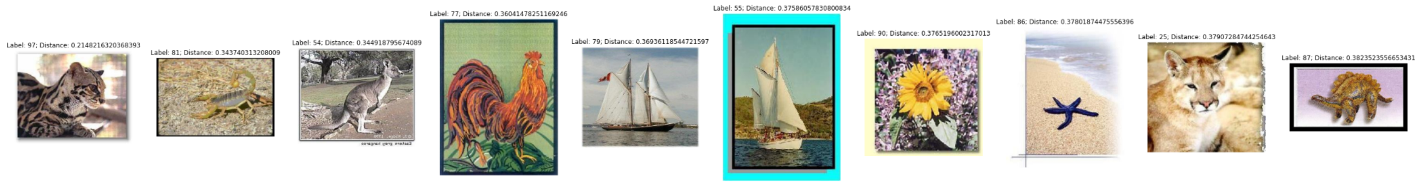

Label: 97; Distance: 0.2148216320368393
Label: 81; Distance: 0.343740313208009
Label: 54; Distance: 0.344918795674089
Label: 77; Distance: 0.36041478251169246
Label: 79; Distance: 0.36936118544721597
Label: 55; Distance: 0.37586057830800834
Label: 90; Distance: 0.3765196002317013
Label: 86; Distance: 0.37801874475556396
Label: 25; Distance: 0.37907284744254643
Label: 87; Distance: 0.3823523556653431

Image index=2500:

Showing 10 similar labels for image ID 2500, using pearson_distance_measure for avgpool_fd feature descriptor…

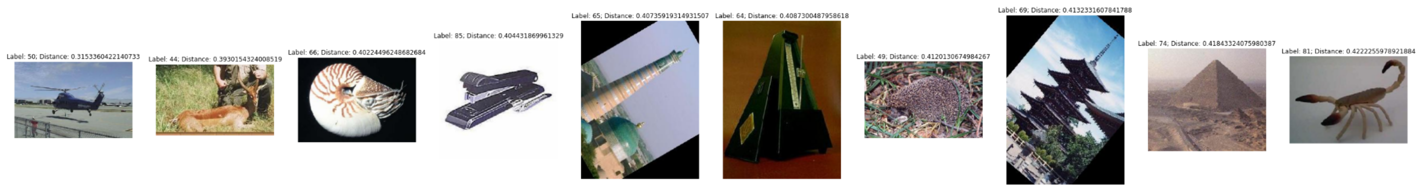
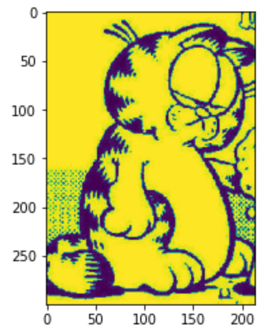

Label: 50; Distance: 0.3153360422140733
Label: 44; Distance: 0.3930154324008519
Label: 66; Distance: 0.40224496248682684
Label: 85; Distance: 0.404431869961329
Label: 65; Distance: 0.40735919314931507
Label: 64; Distance: 0.4087300487958618
Label: 49; Distance: 0.4120130674984267
Label: 69; Distance: 0.4132331607841788
Label: 74; Distance: 0.41843324075980387
Label: 81; Distance: 0.4222255978921884

Image index=5122:



Showing 10 similar labels for image ID 5122, using cosine_distance_measure for hog_fd feature descriptor...

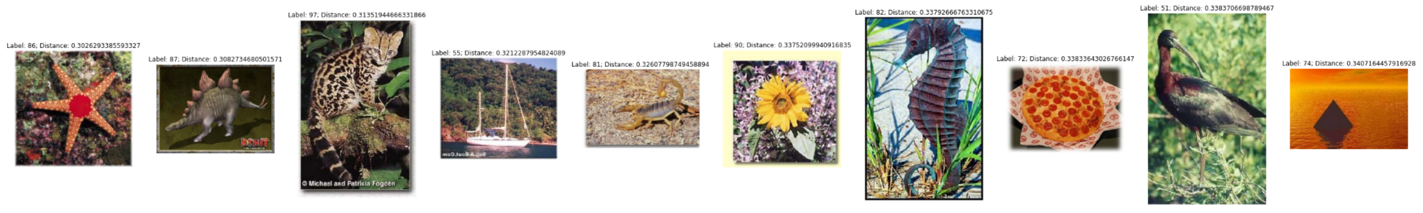
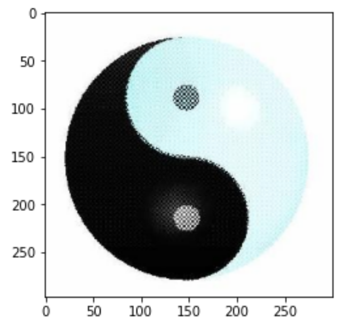
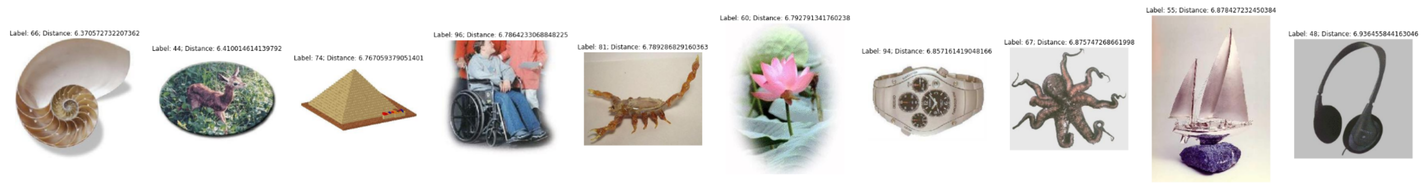

Label: 86; Distance: 0.3026293385593327
Label: 87; Distance: 0.3082734680501571
Label: 97; Distance: 0.31351944666331866
Label: 55; Distance: 0.3212287954824089
Label: 81; Distance: 0.32607798749458894
Label: 90; Distance: 0.33752099940916835
Label: 82; Distance: 0.33792666763310675
Label: 72; Distance: 0.33833643026766147
Label: 51; Distance: 0.3383706698789467
Label: 74; Distance: 0.3407164457916928

Image index=8676:



Showing 10 similar labels for image ID 8676, using euclidean_distance_measure for cm_fd feature descriptor...


Label: 66; Distance: 6.370572732207362
Label: 44; Distance: 6.410014614139792
Label: 74; Distance: 6.767059379051401
Label: 96; Distance: 6.7864233068848225
Label: 81; Distance: 6.789286829160363
Label: 60; Distance: 6.792791341760238
Label: 94; Distance: 6.857161419048166
Label: 67; Distance: 6.875747268661998
Label: 55; Distance: 6.878427232450384
Label: 48; Distance: 6.936455844163046

2B)

Specification:

Given an image ID or select an image file and the value K. It will display k most similar labels according to the Image ID or image file and the feature descriptors obtained by the resnet50 model along with the scores.

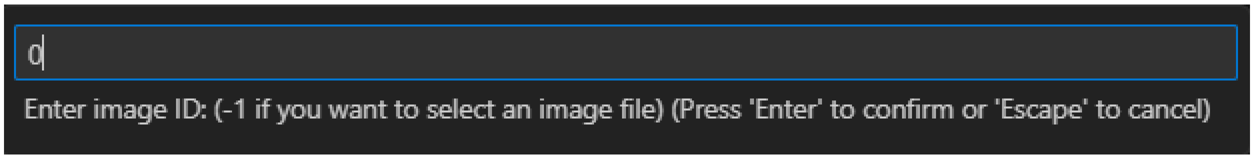
Description:

This program works by first extracting the feature vector from the query image using the resnet50 feature space. It then calculates the similarity between the query image's feature vector and the resnet50 feature vectors of all the images in the database. The k labels with the highest similarity scores are then returned as output.

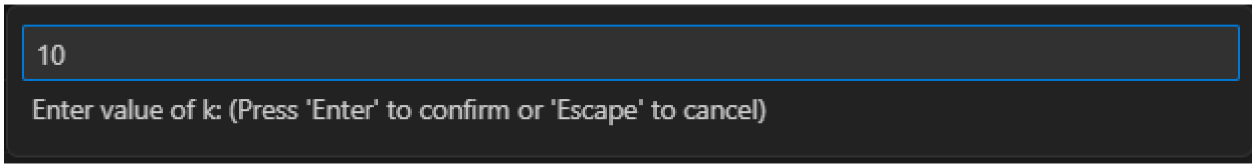
Design:

1. In this task, the resnet50 model is run to conclusion without any hooks at the intermediate layers, the features obtained upon running the entire model are computed and stored into the existing MongoDB database.
2. The image ID entered is searched for in the database, if not found the feature descriptors are calculated. If the input is an image file, all the values except the true_label value are determined.
3. Two dictionaries min_dists and label_dict are initialized in which the 'key' is the image ID and the 'value' is the similarity distance and label for min_dists and label_dict respectively.
4. Using the feature vectors and based on the type of feature space, the first k-most similar images with unique (non-repeated) labels are determined by simultaneously checking for the label value of the similar image.
5. The similarity is determined for all the images and the ones with even better similarity scores replace the ones with the worst similarity scores in both the dictionaries, simultaneously checking for non-repeated labels.
6. The min_dists dictionary is finally sorted in ascending order and the output is displayed with the label and its corresponding similarity value,
7. During this process the images with the same label as the input image are not compared.

Input:



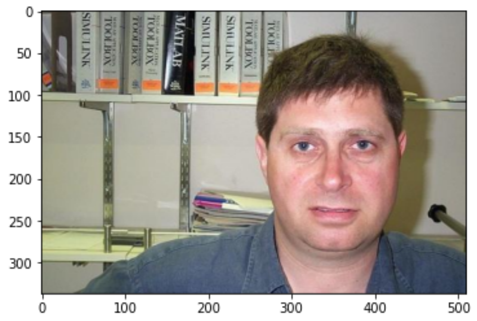
```
0
Enter image ID: (-1 if you want to select an image file) (Press 'Enter' to confirm or 'Escape' to cancel)
```



```
10
Enter value of k: (Press 'Enter' to confirm or 'Escape' to cancel)
```

Output:

Image index=0:



Showing 10 similar labels for image ID 0, using pearson_distance_measure for resnet_fd feature descriptor...

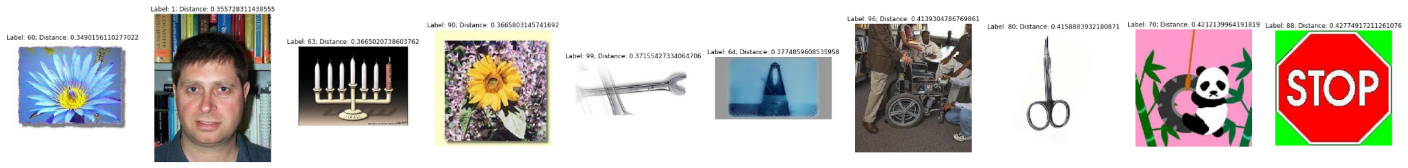
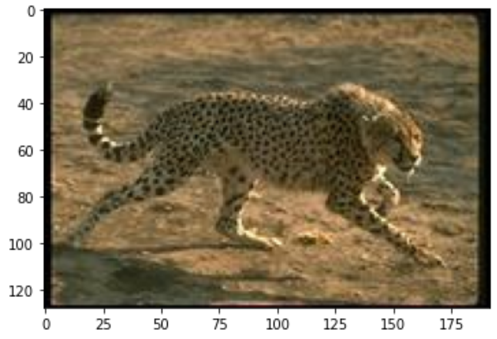

Label: 60; Distance: 0.3490156110277022
Label: 1; Distance: 0.355728311438555
Label: 63; Distance: 0.3665020738603762
Label: 90; Distance: 0.3665803145741692
Label: 99; Distance: 0.37155427334064706
Label: 64; Distance: 0.3774859608535958
Label: 96; Distance: 0.4139304786769861
Label: 80; Distance: 0.4158883932180871
Label: 70; Distance: 0.4212139964191819
Label: 88; Distance: 0.42774917211261076
STOP

Image index=880:



Showing 10 similar labels for image ID 880, using pearson_distance_measure for resnet_fd feature descriptor...

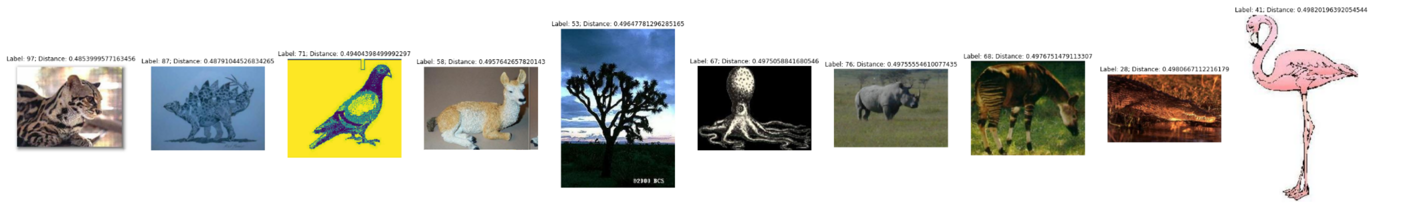
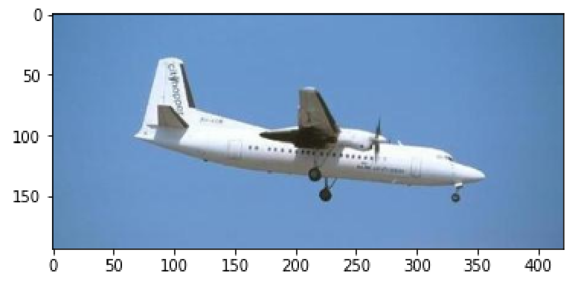

Label: 97; Distance: 0.4853999577163456
Label: 87; Distance: 0.48791044526834265
Label: 71; Distance: 0.49404398499992297
Label: 58; Distance: 0.4957642657820143
Label: 53; Distance: 0.49647781296285165
Label: 67; Distance: 0.4975058841680546
Label: 76; Distance: 0.49755554610077435
Label: 68; Distance: 0.4976751479113307
Label: 28; Distance: 0.4980667112216179
Label: 41; Distance: 0.49820196392054544

Image index=2500:



Showing 10 similar labels for image ID 2500, using pearson_distance_measure for resnet_fd feature descriptor…

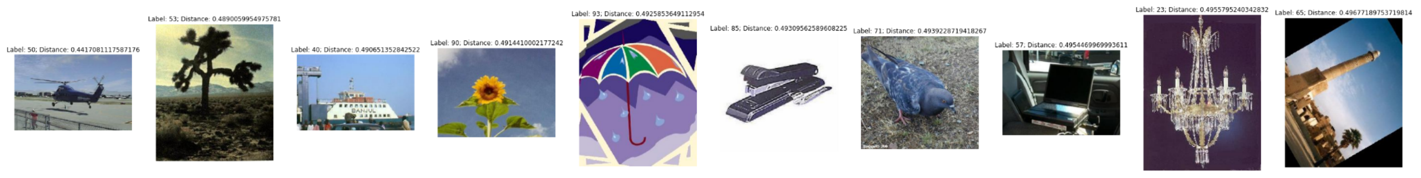
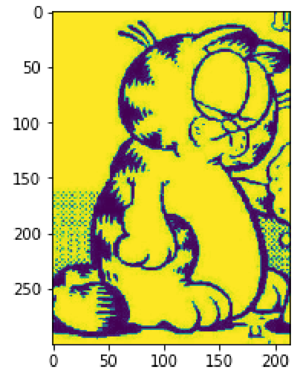

Label: 50; Distance: 0.4417081117587176
Label: 53; Distance: 0.4890059954975781
Label: 40; Distance: 0.490651352842522
Label: 90; Distance: 0.4914410002177242
Label: 93; Distance: 0.4925853649112954
Label: 85; Distance: 0.49309562589608225
Label: 71; Distance: 0.4939228719418267
Label: 57; Distance: 0.4954469969993611
Label: 23; Distance: 0.4955795240342832
Label: 65; Distance: 0.49677189753719814

Image index=5122:



Showing 10 similar labels for image ID 5122, using pearson_distance_measure for resnet_fd feature descriptor...

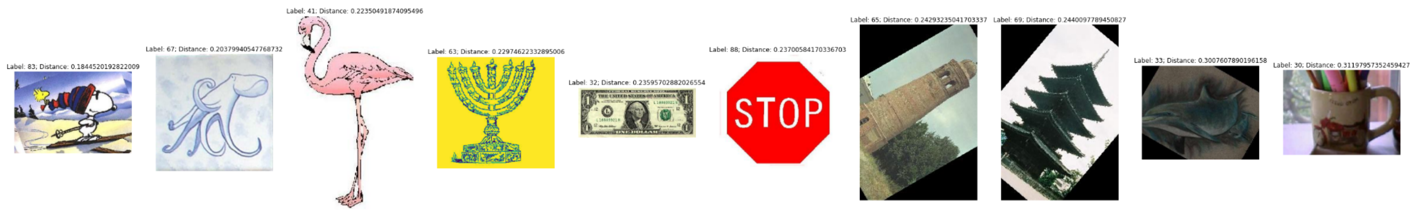
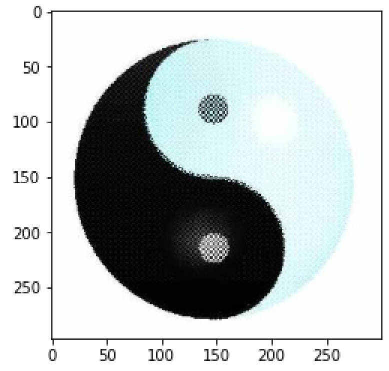
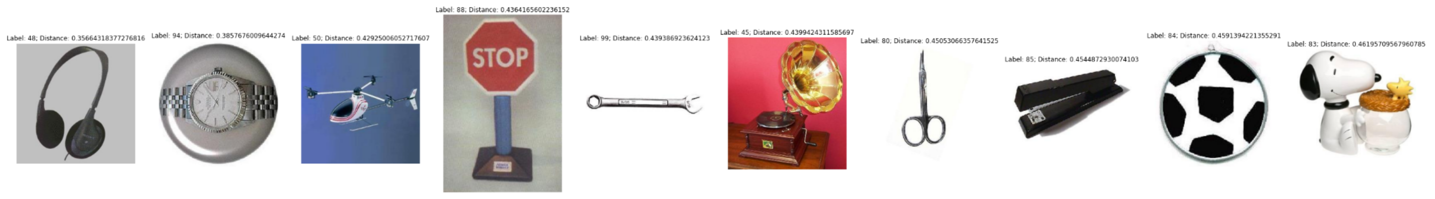

Label: 83; Distance: 0.1844520192822009
Label: 67; Distance: 0.20379940547768732
Label: 41; Distance: 0.22350491874095496
Label: 63; Distance: 0.22974622332895006
Label: 32; Distance: 0.23595702882026554
STOP
Label: 88; Distance: 0.23700584170336703
Label: 65; Distance: 0.24293235041703337
Label: 69; Distance: 0.2440097789450827
Label: 33; Distance: 0.3007607890196158
Label: 30; Distance: 0.31197957352459427

Image index=8676:



Showing 10 similar labels for image ID 8676, using pearson_distance_measure for resnet_fd feature descriptor...


Label: 48; Distance: 0.35664318377276816
Label: 94; Distance: 0.3857676009644274
Label: 50; Distance: 0.42925006052717607
Label: 88; Distance: 0.4364165602236152
STOP
Label: 99; Distance: 0.439386923624123
Label: 45; Distance: 0.4399424311585697
Label: 80; Distance: 0.45053066357641525
Label: 85; Distance: 0.4544872930074103
Label: 84; Distance: 0.4591394221355291
Label: 83; Distance: 0.46195709567960785

TASKS 3-6 (Extraction of latent semantics:

- Note: n - no. of images, k - no. of latent semantics, m - no. of image features
- SVD - Performed using eigendecomposition to get a diagonal matrix of singular values and factor matrices of the singular value's corresponding singular vectors. Latent semantics are stored as "image-semantic" - left factor matrix (nxk), "semantics-core" - singular values (kxk), "semantics-feature" - transpose of right factor matrix (kxm)
- NNMF - Performed using multiplicative update method to get factor matrices that approximate the original data matrix. Latent semantics are stored as "image-semantic" - W matrix (nxk), "semantics-feature" - H matrix (kxm).
- LDA - Performed using scikit-learn's LatentDirichletAllocation class functions to get topic distribution in documents and word distribution in topics. Not particularly suited for continuous data as used in this project, since the dictionary set needed for this algorithm would grow to be very large and can be very inefficient, but can still work nonetheless. Latent semantics are stored as "image-semantic" - word-topic distribution (nxk), "semantics-feature" - topic-document distribution (kxm). The LDA model is also stored to store all the parameters involved for reconstruction in later tasks.
- K-means clustering - Performed iteratively by finding means of clusters and readjusting cluster centers, using Euclidean distance measure. Latent semantics are stored as "image-semantic" - distances of data points from centroids (nxk), "semantics-feature" - centroids' coordinates in original feature space.

TASK 3

Specification:

The user is required to input one of the feature models, the value k and one of the four dimensionality reduction techniques (SVD, NNMF, LDA, k-means). The program calculates a list of the top-k latent semantics extracted under the selected feature space, stored in a properly named output file, and also displays the imageID-weight pairs sorted by weight for each latent semantic

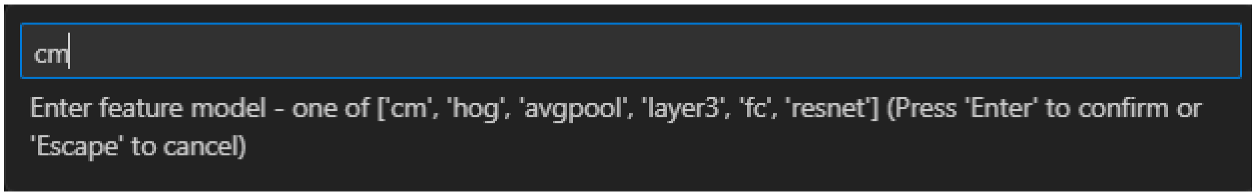
Description:

In this task, the program displays the top-k image-weight pairs based on the feature model and dimensionality reduction technique selected by the user. The program works by extracting the features of the images based on the selected feature space and applies dimensionality reduction technique selected by the user and displays the output accordingly. Modifications according to the functions and its requirements are made to the input in order to process it and display the desired result.

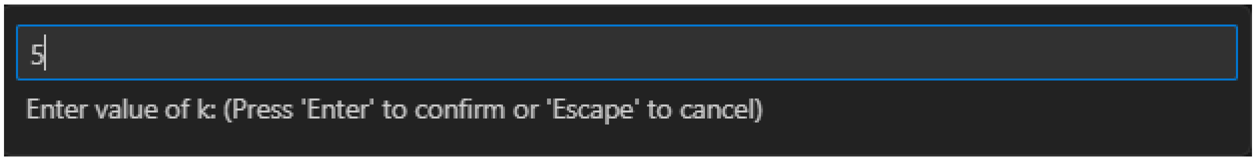
Design:

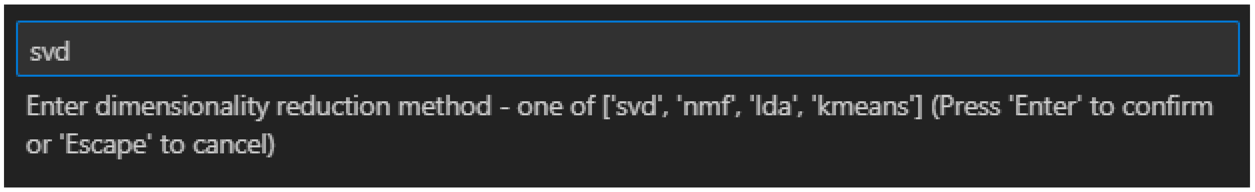
1. The data of all the images in the database are saved in all_images list and then this list is iterated to extract the image_id of all the images and is then saved in feature_ids
2. The feature vectors of the specified model are extracted and flattened and are stored in feature_vectors array
3. For each method, the corresponding dimensionality reduction technique is applied and the latent semantics are calculated and stored in all_latent_semantics dictionary.
    a. For NNMF and LDA the input data is adjusted by shifting the values to ensure non-negativity.
    b. In the case of K-Means clustering, the code stores distances instead of weights and displays them in ascending order.
4. For each latent semantic the imageID-weight pairs are sorted by weights in decreasing order and are displayed
5. The latent semantics are saved in a JSON file
    a. In the case of LDA, the corresponding LDA model is also saved to a .joblib file

Input:



```
cm
Enter feature model - one of ['cm', 'hog', 'avgpool', 'layer3', 'fc', 'resnet'] (Press 'Enter' to confirm or 'Escape' to cancel)
```


```
5
Enter value of k: (Press 'Enter' to confirm or 'Escape' to cancel)
```


```
svd
Enter dimensionality reduction method - one of ['svd', 'nmf', 'lda', 'kmeans'] (Press 'Enter' to confirm or 'Escape' to cancel)
```

Output:

feature space {CM}; k =5; dim.red. {SVD}

```
Applying svd on the cm_fd space to get 5 latent semantics (showing only top 10 image-weight pairs for each latent semantic)...
Latent semantic no. 0
Image_ID    8570  -    Weight  0.08379938013632154
Image_ID    7784  -    Weight  0.07238472588049127
Image_ID    4152  -    Weight  0.060769224719766424
Image_ID    5114  -    Weight  0.05387212151769047
Image_ID    7774  -    Weight  0.05324887247523999
Image_ID    8614  -    Weight  0.05319742868629019
Image_ID    3072  -    Weight  0.0508399452179281
Image_ID    7798  -    Weight  0.050598074135948995
Image_ID    5118  -    Weight  0.05022770477320986
Image_ID    7040  -    Weight  0.04996996742218061
Latent semantic no. 1
Image_ID    8570  -    Weight  0.0708242114969575
Image_ID    7774  -    Weight  0.0654659454748679
….
….
Image_ID    1788  -    Weight  0.04089406629514225
Image_ID    1796  -    Weight  0.040688152223479164
Latent semantic no. 4
Image_ID    8582  -    Weight  0.02577153311253714
Image_ID    8612  -    Weight  0.025608143819276414
Image_ID    7290  -    Weight  0.025578071187110515
Image_ID    7298  -    Weight  0.02535046780104084
Image_ID    7302  -    Weight  0.025316611409381136
Image_ID    7318  -    Weight  0.02521277976701421
Image_ID    8580  -    Weight  0.025201323062899246
Image_ID    6392  -    Weight  0.025170862056424635
Image_ID    2738  -    Weight  0.0251065168979951
Image_ID    6420  -    Weight  0.02510499876667636
```

feature space {RESNET}; k =5; alt. dim.red. {LDA}

```
Applying lda on the resnet_fd space to get 5 latent semantics (showing only top 10 image-weight pairs for each
latent semantic)...
iteration: 1 of max_iter: 10
iteration: 2 of max_iter: 10
….
….
iteration: 10 of max_iter: 10
Latent semantic no. 0
Image_ID    232     -    Weight  0.527624883934027
Image_ID    294     -    Weight  0.4935692184445949
Image_ID    238     -    Weight  0.4663024286585402
Image_ID    832     -    Weight  0.44777572628617457
Image_ID    254     -    Weight  0.4429147064629401
Image_ID    522     -    Weight  0.37939812923171684
Image_ID    252     -    Weight  0.3633398995719225
Image_ID    384     -    Weight  0.3626769302117395
Image_ID    492     -    Weight  0.35913152899218626
Image_ID    256     -    Weight  0.345355884928096
Latent semantic no. 1
Image_ID    1202    -    Weight  0.5115083436427273
Image_ID    1830    -    Weight  0.49814167739248916
….
….
Image_ID    8642    -    Weight  0.5976118861485914
Image_ID    2972    -    Weight  0.5976076540556043
Latent semantic no. 4
Image_ID    1202    -    Weight  0.10076460444926877
Image_ID    232     -    Weight  0.10075529766105475
Image_ID    1830    -    Weight  0.10075055706613864
Image_ID    294     -    Weight  0.10074432823596924
Image_ID    1724    -    Weight  0.10072174868127542
Image_ID    1588    -    Weight  0.10072004476930081
Image_ID    8504    -    Weight  0.1007186154815303
Image_ID    1042    -    Weight  0.10071824313734666
Image_ID    1506    -    Weight  0.10071721546078308
Image_ID    238     -    Weight  0.10071721226442397
```

TASK 4

Specification:

The user is required to input one of the feature models and the value k. The program will display a list of the top-k latent semantics extracted using CP-decomposition of a three modal (image-feature-label) tensor under the selected features space, stored in a properly named output file, with each latent semantic presented in the form of a list of label-weight pairs, ordered in decreasing order of weights.

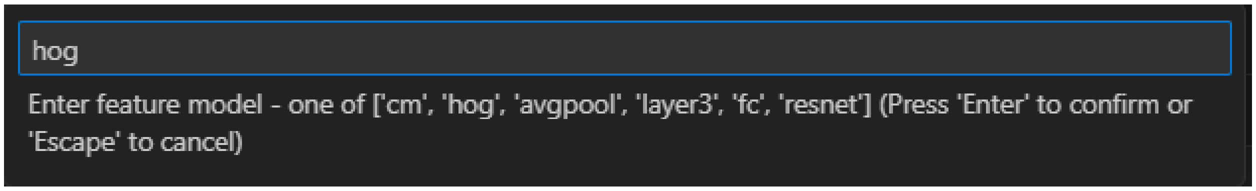
Description:

In this task, the program displays the top-k label-weight pairs obtained by CP decomposition based on the feature model selected by the user. The program works by extracting the features of the images based on the selected feature space and applies CP decomposition to it.

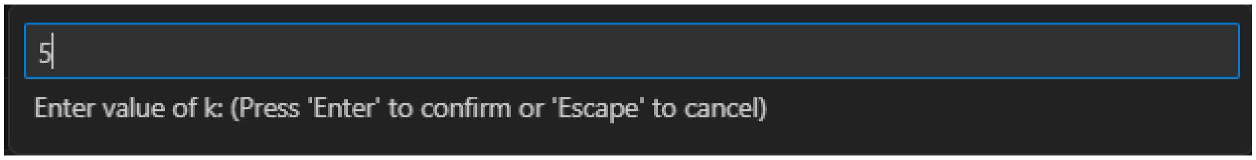
Design:

1. The documents from the database are extracted and stored in a list, all_images. Further, the image IDs of all the images (documents) in the database are obtained and stored in another list, img_ids.
2. Img_feature_ids list is declared in order to store the indices of all the features. The img_label_ids list extracts the labels from the database. Further, in another list feature_ids, img_ids, img_feature_ids, img_label_ids lists are combined.
3. CP decomposition (tensorly library function) is then applied to the data and the weight tensors and the factor matrices are determined
    a. A function compute_cp_decomposition is defined in order to calculate the CP decomposition for the data given.
    b. The data is retrieved from the database and stored in all_images list.
    c. For each image (document), the label associated with the image is retrieved and the feature vector associated with the image is flattened. This data is stored in data_tensor
    d. CP decomposition is applied to data_tensor using the tensorly.decomposition.parafac function. The rank parameter is specified, which indicates the number of latent components to extract. The normalize_factors parameter is set to True, which means the factor matrices are normalized during the decomposition.
    e. The function returns the weights_tensor and factor_matrices as the result of the CP decomposition.
4. The latent semantics are stored in the all_latent_semantics dictionary which includes "image-semantic," "feature-semantic," "label-semantic," and "semantics-core."
5. The extracted latent semantics are sorted and displayed for: "image," "feature," and "label." It shows object IDs (e.g., image, feature, label) and their associated weights in descending order.
6. Finally, the function saves the extracted latent semantics in a JSON file with a name that reflects the chosen feature model, rank, and the term "semantics."

Input:


hog

Enter feature model - one of ['cm', 'hog', 'avgpool', 'layer3', 'fc', 'resnet'] (Press 'Enter' to confirm or 'Escape' to cancel)


5

Enter value of k: (Press 'Enter' to confirm or 'Escape' to cancel)

Output:

feature space {HOG}; k =5

Applying CP decomposition on the hog_fd space to get 5 latent semantics (showing only top 10 image-weight pairs for each latent semantic)...

```
(4339, 900, 101)
Showing image-weight latent semantic
Latent semantic no. 0
image   907   -   weight   0.08111146004467022
image   816   -   weight   0.07592488869987997
image   735   -   weight   0.07458297701723804
image   826   -   weight   0.07380720903908039
image   900   -   weight   0.07254260686913609
image   853   -   weight   0.07232350843307524
image   880   -   weight   0.07216559248117042
...
…
…
Latent semantic no. 4
image   99    -   weight   0.12343012089910918
image   119   -   weight   0.113955175449094
image   126   -   weight   0.11296824287606817
image   209   -   weight   0.11265964717862482
image   10    -   weight   0.11155714713370057
image   192   -   weight   0.10955555395796271
image   101   -   weight   0.10818386615418286
image   160   -   weight   0.10667367518297338
image   118   -   weight   0.10531173438164543
image   127   -   weight   0.10394753636656369
Showing feature-weight latent semantic
Latent semantic no. 0
feature 8     -   weight   0.007549630754166452
feature 2     -   weight   0.005954238672423357
feature 9     -   weight   0.004876533386956757
feature 0     -   weight   0.0034910158424980285
feature 6     -   weight   0.0022185874467824103
feature 4     -   weight   0.0018026862802200542
feature 1       -       weight  0.001068830578501311
feature 7       -       weight  0.0009374842411715104
feature 3       -       weight  0.0008517960973485152
feature 5       -       weight  0.0004925790177787824
Latent semantic no. 1
feature 8       -       weight  0.04963851572587758
feature 9       -       weight  0.02469073501264152
…
…
…

feature 8       -       weight  0.02561896078346682
feature 9       -       weight  0.01319754040149297
feature 7       -       weight  0.012705985143148904
feature 0       -       weight  0.010993180206407056
feature 5       -       weight  0.010552070705402256
feature 1       -       weight  0.010362460432408627
feature 3       -       weight  0.00967822121416682
Showing label-weight latent semantic
Latent semantic no. 0
label   3       -       weight  0.9999999999999998
label   0       -       weight  0.0
label   1       -       weight  0.0
label   2       -       weight  0.0
label   4       -       weight  0.0
label   5       -       weight  0.0
label   6       -       weight  0.0
…
…
…
label   4       -       weight  0.0
label   5       -       weight  0.0
label   6       -       weight  0.0
label   7       -       weight  0.0
label   8       -       weight  0.0
label   9       -       weight  0.0
```

feature space {RESNET}; k =5

```
Applying CP decomposition on the resnet_fd space to get 5 latent semantics (showing only top 10 image-weight
pairs for each latent semantic)...
(4339, 1000, 101)
Showing image-weight latent semantic
Latent semantic no. 0
image   1290    -       weight  0.0847666356145715
image   1129    -       weight  0.08242073414148895
image   1241    -       weight  0.0799569904030165
image   1027    -       weight  0.07888186233401384
image   1032    -       weight  0.07819874598713544
image    1146    -    weight   0.07783171600166511
image    1319    -    weight   0.07689988509234584
image    1071    -    weight   0.07607247416023259
image    1149    -    weight   0.07491172495114765
image    1168    -    weight   0.07466720796517286
Latent semantic no. 1
image    601     -    weight   0.10862382857157002
…
…
…
Latent semantic no. 4
image    4152    -    weight   0.24625882198147067
image    4138    -    weight   0.21401374422251226
image    4115    -    weight   0.18803437036733134
image    4118    -    weight   0.1869596796211395
image    4088    -    weight   0.18401510782266442
image    4120    -    weight   0.17782839590955185
image    4144    -    weight   0.17605394821662454
image    4093    -    weight   0.17602132979364954
image    4140    -    weight   0.1734184098438752
image    4169    -    weight   0.1731929013678939
Showing feature-weight latent semantic
Latent semantic no. 0
feature  0       -    weight   0.0011997027067116015
Latent semantic no. 1
feature  0       -    weight   0.0015211236139400807
…
…
…

Latent semantic no. 4
label    94      -    weight   1.0000000000000002
label    0       -    weight   0.0
label    1       -    weight   0.0
label    2       -    weight   0.0
label    3       -    weight   0.0
label    4       -    weight   0.0
label    5       -    weight   0.0
label    6       -    weight   0.0
label    7       -    weight   0.0
label    8       -    weight   0.0
```

TASK 5

Specification:

The user is required to input one of the feature models and the value k. The program calculates a label-label similarity matrix, list of the latent semantics extracted using the selected dimensionality reduction technique on the label-label similarity matrix and list of label-weight pairs, ordered in decreasing order of weights

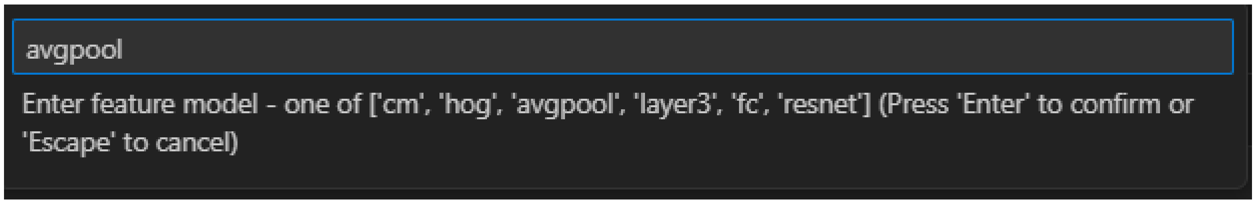
Description:

In this task, the program computes the label-label similarity matrix based on the feature model selected by the user. This is done by calculating the similarity between each pair of labels using the selected feature model. Further, dimensionality reduction technique selected by the user is applied and the latent semantics are extracted which then gives the top-k list of label-weight pairs, which . Modifications according to the functions and its requirements are made to the input in order to process it and display the desired result.

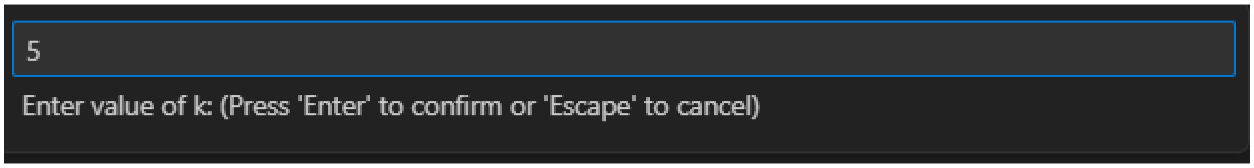
Design:

1. First, the label-label similarity matrix is calculated.
    a. The mean label representative vector for each label based on the feature space selected by the user is calculated and stored in the label_mean_vectors list.
    b. Another list label_sim_matrix is initialized to zero, this list stores the label similarity matrix. This matrix will store the pairwise label similarities
    c. The label similarities for all unique label pairs are calculated. The lower triangular part of the matrix to calculate the similarities and the upper triangular part is filled based on symmetry.
    d. The similarity between the representative vectors for the labels is calculated using distance function. The calculated similarity values are stored in the label_sim_matrix.
2. Next, the latent semantics are computed from the similarity. For each method, the corresponding dimensionality reduction technique is applied and the latent semantics are calculated and stored in all_latent_semantics dictionary
    a. For SVD, the latent semantics are organized into lists of label-semantic pairs.
    b. For NNMF and LDA the input data is adjusted by shifting the values to ensure non-negativity.
    c. In the case of K-Means clustering, the code stores distances instead of weights and displays them in ascending order.
    d. For each latent semantic the label-weight pairs are sorted by weights in decreasing order and are displayed
    e. The latent semantics are saved in a JSON file
        i. In the case of LDA, the corresponding LDA model is also saved to a .joblib file

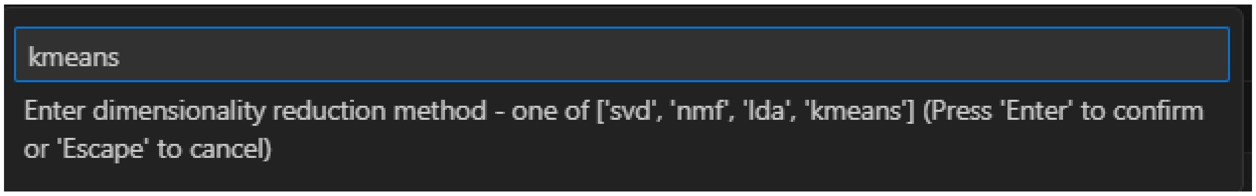
Input:



```
avgpool
Enter feature model - one of ['cm', 'hog', 'avgpool', 'layer3', 'fc', 'resnet'] (Press 'Enter' to confirm or 'Escape' to cancel)
```



```
5
Enter value of k: (Press 'Enter' to confirm or 'Escape' to cancel)
```



```
kmeans
Enter dimensionality reduction method - one of ['svd', 'nmf', 'lda', 'kmeans'] (Press 'Enter' to confirm or 'Escape' to cancel)
```

Output:

feature space {AvgPool}; k=5; dim.red. {kmeans}

```
Applying kmeans on the given similarity matrix to get 5 latent semantics (showing only top 10 label-weight pairs for each latent semantic)...
Initialized centroids
Iteration 4 - Converged
Note: for K-Means we display distances, in ascending order
Latent semantic no. 0
label	90	-	Distance 0.32892272329969746
label	60	-	Distance 0.3775083857073132
label	95	-	Distance 0.39926812208093315
label	89	-	Distance 0.4246002553006171
label	72	-	Distance 0.5195317997904244
label	30	-	Distance 0.5409102035047411
label	16	-	Distance 0.6180309021389221
label	13	-	Distance 0.6308101616713481
label	70	-	Distance 0.6650708816205688
label	39	-	Distance 0.6654131462823251
…
…
…
label	53	-	Distance 0.5411225199375898
label	68	-	Distance 0.5419292240863035
label	2	-	Distance 0.5533024079373133
label	49	-	Distance 0.5651531309897075
label	54	-	Distance 0.5744378860573579
label	44	-	Distance 0.5764839029894493
```

feature space {RESNET}; k=5; dim.red. {kmeans}

```
Applying kmeans on the given similarity matrix to get 5 latent semantics (showing only top 10 label-weight pairs for
each latent semantic)...
Initialized centroids
Iteration 1 - Converged
Note: for K-Means we display distances, in ascending order
Latent semantic no. 0
label   56      -       Distance 0.5078837376965605
label   100     -       Distance 0.5082944283737307
label   63      -       Distance 0.5109584766064246
label   30      -       Distance 0.5125112618608927
label   16      -       Distance 0.5152144525896714
label   39      -       Distance 0.5152886797665173
label   23      -       Distance 0.546036781260073
label   15      -       Distance 0.5490677214673996
label   12      -       Distance 0.5570006656832791
label   13      -       Distance 0.5592056241258636
…
…
…
Latent semantic no. 4
label   87      -       Distance 0.10694843283352252
label   14      -       Distance 0.10694843283352254
label   76      -       Distance 0.6210334618347675
label   36      -       Distance 0.675496460846602
label   68      -       Distance 0.7608216345974752
label   53      -       Distance 0.7911539612020739
label   37      -       Distance 0.8292263011970864
label   10      -       Distance 0.8360058965954094
label   9       -       Distance 0.8394108966977034
label   7       -       Distance 0.8405579416874938
```

TASK 6

Specification:

The user is required to input one of the feature models and the value k. It will output an image-image similarity matrix, list of the latent semantics extracted using the selected dimensionality reduction technique on the image-image similarity matrix and list of image-weight pairs, ordered in decreasing order of weights

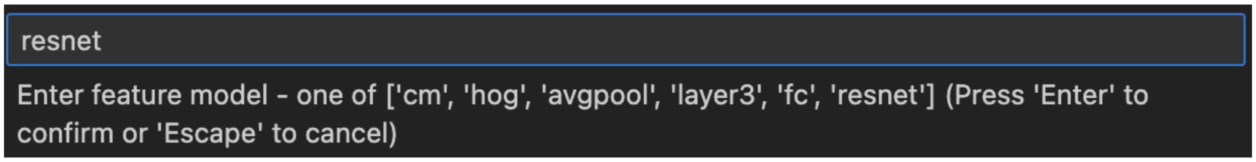
Description:

In this task, the program computes the image-image similarity matrix based on the feature model selected by the user. This is done by calculating the similarity between each pair of images using the selected feature model. Further, dimensionality reduction technique selected by the user is applied and the latent semantics are extracted which then gives the top-k list of label-weight pairs, which .

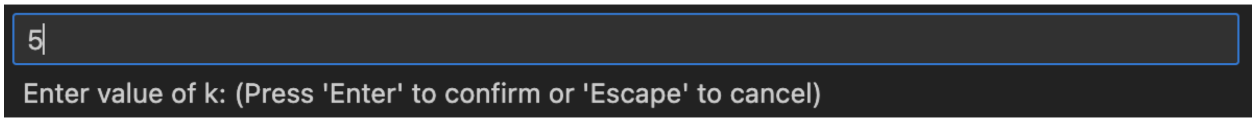
Design:

1. First, the image-image similarity matrix is calculated.
    a. The feature vector for the feature model is extracted from the database.
    b. Another list label_sim_matrix is initialized to zero, this list stores the label similarity matrix. This matrix will store the pairwise image similarities
    c. The label similarities for all unique image pairs are calculated. The lower triangular part of the matrix to calculate the similarities and the upper triangular part is filled based on symmetry.
    d. The similarity between the vectors of the images is calculated using distance function. The calculated similarity values are stored in the label_sim_matrix.
2. Next, the latent semantics are computed from the similarity. For each method, the corresponding dimensionality reduction technique is applied and the latent semantics is calculated and stored in all_latent_semantics dictionary
    a. For SVD, the latent semantics are organized into lists of image-semantic pairs.
    b. For NNMF and LDA the input data is adjusted by shifting the values to ensure non-negativity.
    c. In the case of K-Means clustering, the code stores distances instead of weights and displays them in ascending order.
    d. For each latent semantic the label-weight pairs are sorted by weights in decreasing order and are displayed
    e. The latent semantics are saved in a JSON file
        i. In the case of LDA, the corresponding LDA model is also saved to a .joblib file

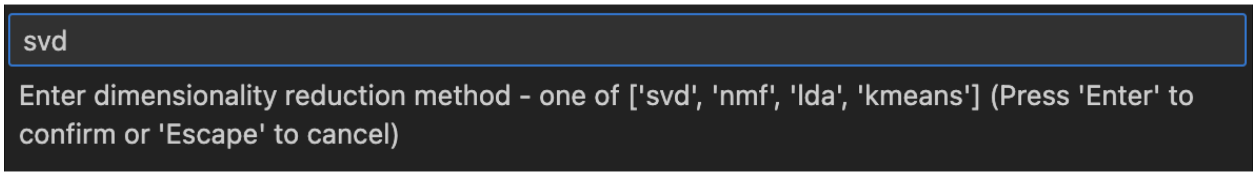
Input:



```
resnet
Enter feature model - one of ['cm', 'hog', 'avgpool', 'layer3', 'fc', 'resnet'] (Press 'Enter' to confirm or 'Escape' to cancel)
```



```
5
Enter value of k: (Press 'Enter' to confirm or 'Escape' to cancel)
```



```
svd
Enter dimensionality reduction method - one of ['svd', 'nmf', 'lda', 'kmeans'] (Press 'Enter' to confirm or 'Escape' to cancel)
```

Output:

feature space {L3}; k=5; dim.red. {SVD}

```
Applying svd on the given similarity matrix to get 5 latent semantics (showing only top 10 image-weight pairs for
each latent semantic)...
Latent semantic no. 0
image   1491    -       Weight  (0.016513738690806405+0j)
image   2817    -       Weight  (0.016471280632184785+0j)
image   1816    -       Weight  (0.0164533713414888+0j)
image   1811    -       Weight  (0.016429697005778243+0j)
image   2868    -       Weight  (0.016422997018732444+0j)
image   1763    -       Weight  (0.016415099339543864+0j)
image   1820    -       Weight  (0.016397607052868645+0j)
image   2885    -       Weight  (0.01638372338832298+0j)
image   1822    -       Weight  (0.016376095934206268+0j)
image   1768    -       Weight  (0.016360633549188486+0j)
…
…
…

image   2997    -       Weight  (0.026259464147068083+0j)
image   2872    -       Weight  (0.026246697458263493+0j)
Latent semantic no. 4
image   3746    -       Weight  (0.04791648677233576+0j)
image   3290    -       Weight  (0.047349636926180494+0j)
image   3286    -       Weight  (0.04644615072972975+0j)
image   2566    -       Weight  (0.045938692161819+0j)
image   3283    -       Weight  (0.045633426936504184+0j)
image   3263    -       Weight  (0.045022018921337906+0j)
image   1364    -       Weight  (0.04453624264619371+0j)
image   3249    -       Weight  (0.04437297827414635+0j)
image   3494    -       Weight  (0.043527944345823105+0j)
image   1844    -       Weight  (0.04340458352740871+0j)
```

TASK 7

Specification:

User is required to input the latent semantic of the choice, and the specification for that semantic, namely feature model, K value, dimensionality reduction technique, followed by image ID and then K2 value denoting the number of similar objects to find. The program will output the list of K2 most similar images.

Description:

The program works by first extracting the feature vector from the query image. This feature descriptor is then converted from an m dimensional vector to a K dimensional vector. Then we find similarity between this K dimensional vector and the other K dimensional vectors that were extracted during the latent semantic calculations. The K2 images with the least euclidean distance are shown to the user.

Design:

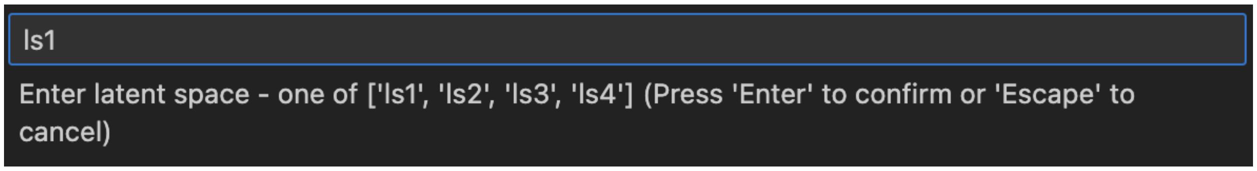
1. Inputs are obtained, and the latent semantics are loaded from the corresponding file.
2. Similar images are found based on the user input for latent semantic based on various cases as follows:
    a. For LS2
        i. Image-semantic matrix and semantic-core from the cp-decomposition output is loaded into the memory
        ii. The image-semantic matrix is multiplied by the semantic-core matrix, which will give us our comparison-feature-space.
        iii. We pick the vector from this comparison_feature_space corresponding to our image, and call this comparison_vector
        iv. We calculate the euclidean distance between this and the other vectors in the comparison space, and display the k most similar images, i.e. k images with the least distance
    b. For LS1, LS3, LS4
        i. Latent semantics are loaded into the memory. A comparison_vector and comparison_feature_space is formed based on the dimensionality reduction method selected.
            1. SVD
                a. Image-semantic (left factor), semantic-core (singular values) and transpose of semantic-feature (right factor) matrices are stored into U, S and V respectively.
                b. comparison_feature_space is calculated by multiplying U and S.
                c. For LS1, Image feature descriptor is fetched from the database and multiplied with V and then S to convert the m dimensional feature vector to K dimensional vector, which will be our comparison vector.
                d. For LS4, comparison_vector is picked from the comparison_feature_space itself by indexing, since we have no way of transforming a m dimensional feature descriptor to NUM_IMAGES dimensional vector
                e. For LS3, since the latent semantic is based on image-image similarity, the label for the input image is fetched from the

database, and the corresponding row from the comparison_feature_space is used as a comparison_vector.

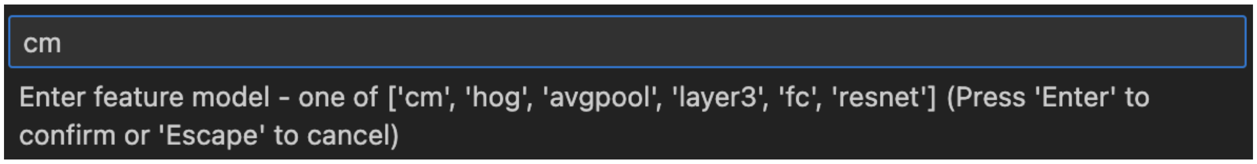
2. NMF
    a. Image-semantic is the comparison_feature_space for LS1 and LS3, and semantic-feature matrix is stored in H
    b. For LS1, comparison_vector is formed by running NMF on it along with the H value stated above, without updating H.
    c. LS3 and LS4 are the same as for SVD
3. K-means
    a. Image-semantic is stored as comparison_feature_space, semantic-feature is stored in S. This S contains the coordinates of the centroids in the original feature space
    b. For LS1, The image feature descriptor is fetched from the database, its distance from each of the K centroids is calculated. The vector of these distances will be the comparison_vector.
    c. LS3 and LS4 are the same as above
4. LDA
    a. The LDA model is loaded along with latent semantics.
    b. comparison_feature_space is the image-semantic stored in the latent semantic file.
    c. For LS1, the feature descriptor is fetched from the database and it is passed through the LDA model's transform function to obtain comparison_vector.
    d. For LS3 and LS4, we use the row from the comparison_feature_space as the comparison_vector

3. Euclidean distances are calculated between the comparison_vector and every other row in the comparison_feature_space except the one corresponding to the image ID
4. In case of LDA dimensionality reduction, we use KL-divergence to calculate the similarity.
5. Based on the latent semantic input, the output is shown as follows
    a. For LS1, LS2 and LS4, we pick the top K2 images from the output array and display it to the user
    b. For LS3, we pick the most similar label to the label of input image and find K2 random images with that label

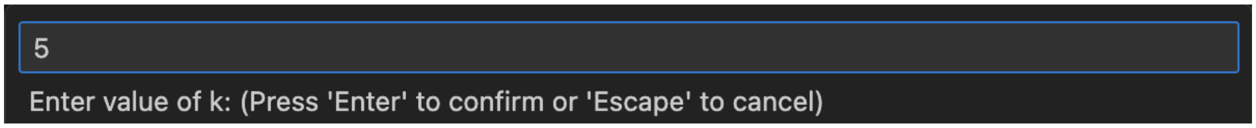
Input:



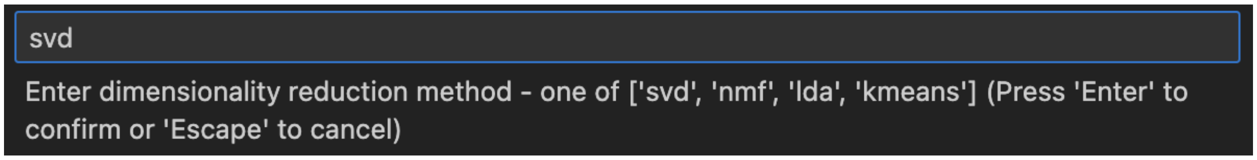
```
ls1
Enter latent space - one of ['ls1', 'ls2', 'ls3', 'ls4'] (Press 'Enter' to confirm or 'Escape' to cancel)
```



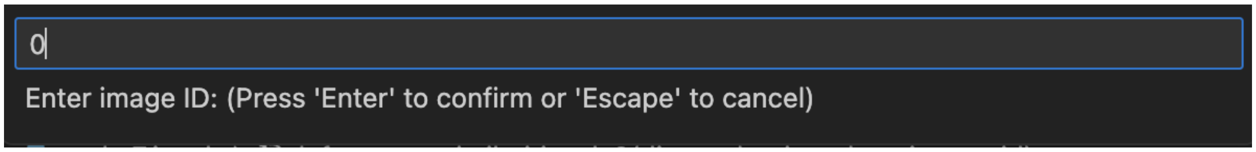
```
cm
Enter feature model - one of ['cm', 'hog', 'avgpool', 'layer3', 'fc', 'resnet'] (Press 'Enter' to confirm or 'Escape' to cancel)
```



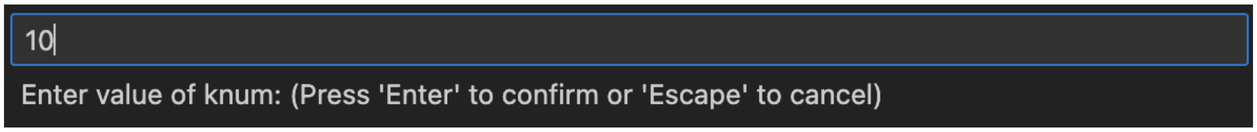
```
5
Enter value of k: (Press 'Enter' to confirm or 'Escape' to cancel)
```



```
svd
Enter dimensionality reduction method - one of ['svd', 'nmf', 'lda', 'kmeans'] (Press 'Enter' to confirm or 'Escape' to cancel)
```



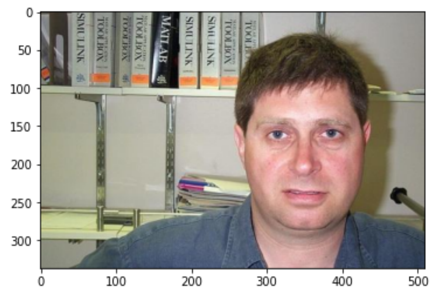
```
0
Enter image ID: (Press 'Enter' to confirm or 'Escape' to cancel)
```



```
10
Enter value of knum: (Press 'Enter' to confirm or 'Escape' to cancel)
```

Output:

image {0}; latent space {T3-CM-5-SVD} ; k=10



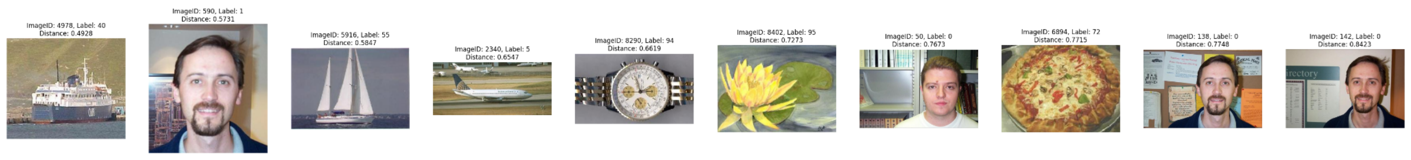
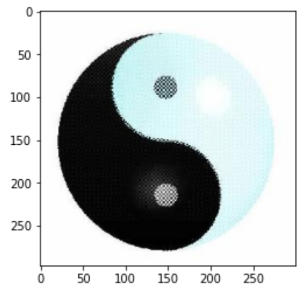
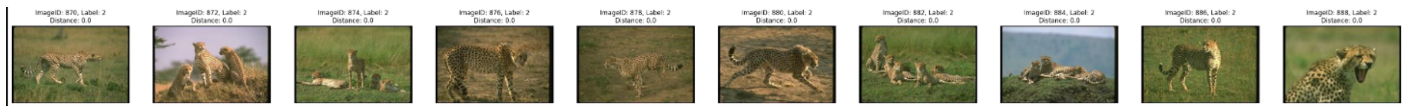


image {8676}; latent space {T4-RESNET-5} ; k=10





TASK 8

Specification:

User is required to input the latent semantic of the choice, and the specification for that semantic, namely feature model, K value, dimensionality reduction technique, followed by image ID and then K2 value denoting the number of similar objects to find. The program will output the list of K2 most similar labels to the input image.

Description:

The program works by first extracting the feature vector from the query image. This feature descriptor is then converted from an m dimensional vector to a K dimensional vector. Then we find similarity between this K dimensional vector and the other K dimensional vectors that were extracted during the latent semantic calculations. The list of K2 most similar labels will be shown to the user.

Design:

1. Inputs are obtained, and the latent semantics are loaded from the corresponding file.
2. Similar images are found based on the user input for latent semantic
   a. For LS2
      i. Image-semantic matrix and semantic-core from the cp-decomposition output is loaded into the memory
      ii. The image-semantic matrix is multiplied by the semantic-core matrix, which will give us our comparison-feature-space.
      iii. A vector is picked from this comparison_feature_space corresponding to our image, which will be our comparison_vector.
      iv. We calculate the euclidean distance between this and the other vectors in the comparison space and then sort these distances in ascending order.
      v. Then, K2 most similar images, each with K2 unique labels are found.

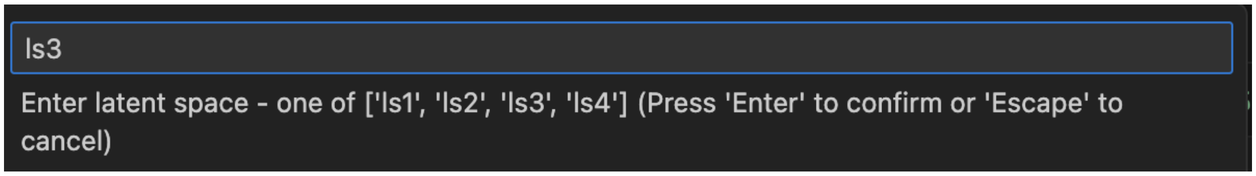
vi. This list of labels is shown to the user.

b. For LS1, LS3, LS4
   i. Latent semantics are loaded into the memory. A comparison_vector and comparison_feature_space is formed based on the dimensionality reduction method selected.
      1. SVD
         a. Image-semantic, semantic-core and semantic-feature matrices are stored into U, S and V respectively.
         b. comparison_feature_space is calculated by multiplying U and S.
         c. For LS1, Image feature descriptor is fetched from the database and multiplied with V and then S to convert the m dimensional feature vector to K dimensional vector, which will be our comparison vector.
         d. For LS4, comparison_vector is picked from the comparison_feature_space itself, since we have no way of transforming a m dimensional feature descriptor ro NUM_IMAGES dimensional vector
         e. For LS3, since the latent semantic is based on image-image similarity, the label for the input image is fetched from the database, and the corresponding row from the comparison_feature_space is used as a comparison_vector.
      2. NMF
         a. Image-semantic is the comparison_feature_space for LS1 and LS3, and semantic-feature is stored in H
         b. For LS1, comparison_vector is formed by running nmf on it along with the H value stated above.
         c. For LS4, comparison_vector is picked from the comparison_feature_space itself.
         d. For LS3, we pick the image-semantic array as the comparison_feature_space. We find the label of the input image and use the corresponding row in the image-semantic as a comparison_vector
      3. K-means
         a. Image-semantic is stored as comparison_feature_space, semantic-feature is stored in S
         b. For LS4, a row from this comparison_feature_space are used as a comparison_vector.
         c. This S contains the coordinates of the centroid in m dimensions
         d. For LS1, The image feature descriptor is fetched from the database, its distance from each of the K centroids is

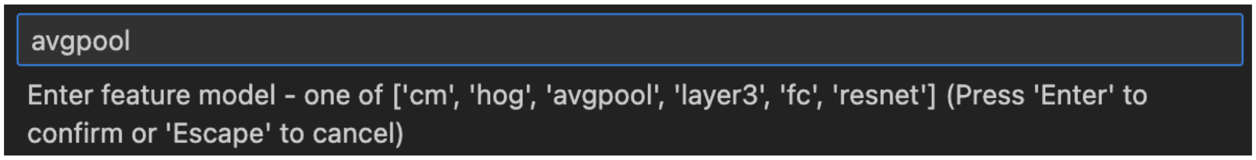
        calculated. The vector of these distances will be the comparison_vector.
        e. For LS3, we use comparison_feature_space row corresponding to the input image label as comparison_vector
    4. LDA
        a. The LDA model is loaded along with latent semantics.
        b. comparison_feature_space is the image-semantic stored in the latent semantic file.
        c. For LS1, the feature descriptor is fetched from the database and it is passed through the LDA model to obtain comparison_vector.
        d. For LS3 and LS4, we use the row from the comparison_feature_space as the comparison_vector

3. Euclidean distances are calculated between the comparison_vector and every other row in the comparison_feature_space except the one corresponding to the image ID
4. In case of LDA dimensionality reduction, we use KL-divergence to calculate the similarity.
5. Based on the latent semantic input, the output is shown as follows
    a. For LS1, LS2 and LS4, we pick the top K2 uniquely labeled images from the output array and display it to the user
    b. For LS3, we pick the K2 labels with least euclidean distances and display them to the user.

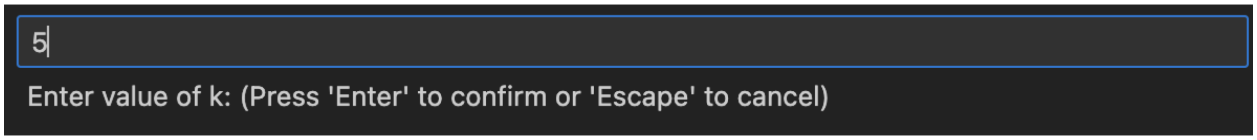
Input:



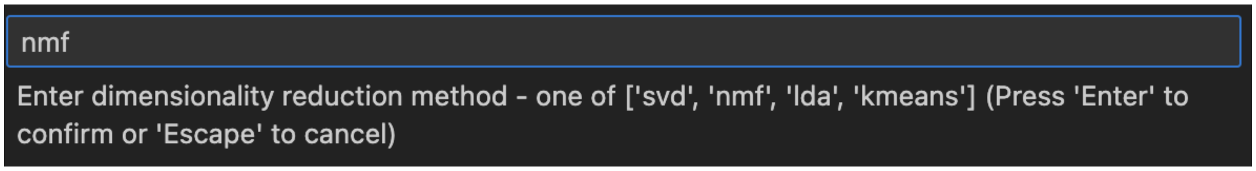
```
ls3
Enter latent space - one of ['ls1', 'ls2', 'ls3', 'ls4'] (Press 'Enter' to confirm or 'Escape' to cancel)
```



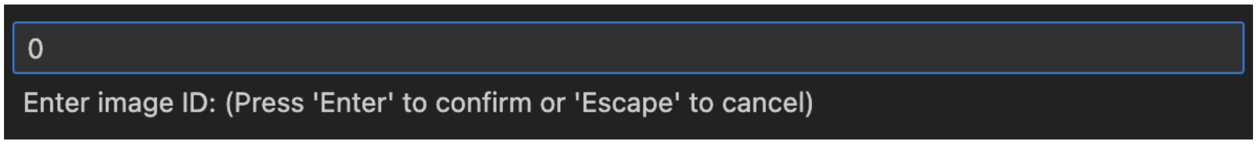
```
avgpool
Enter feature model - one of ['cm', 'hog', 'avgpool', 'layer3', 'fc', 'resnet'] (Press 'Enter' to confirm or 'Escape' to cancel)
```



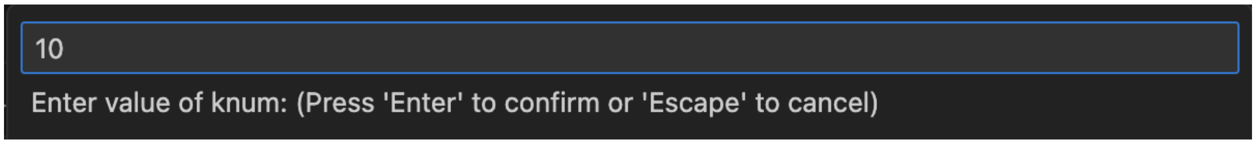
```
5
Enter value of k: (Press 'Enter' to confirm or 'Escape' to cancel)
```



```
nmf
Enter dimensionality reduction method - one of ['svd', 'nmf', 'lda', 'kmeans'] (Press 'Enter' to
confirm or 'Escape' to cancel)
```



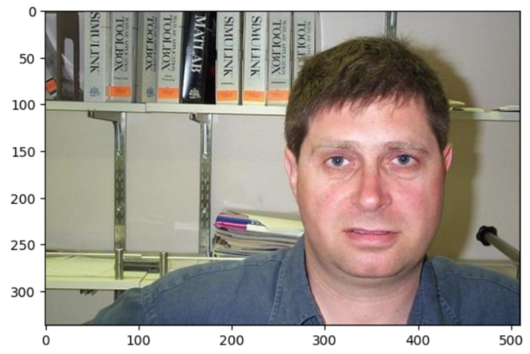
```
0
Enter image ID: (Press 'Enter' to confirm or 'Escape' to cancel)
```



```
10
Enter value of knum: (Press 'Enter' to confirm or 'Escape' to cancel)
```

Output:

image {0}; latent space {T5-avgpool-5-NNMF}; k=10



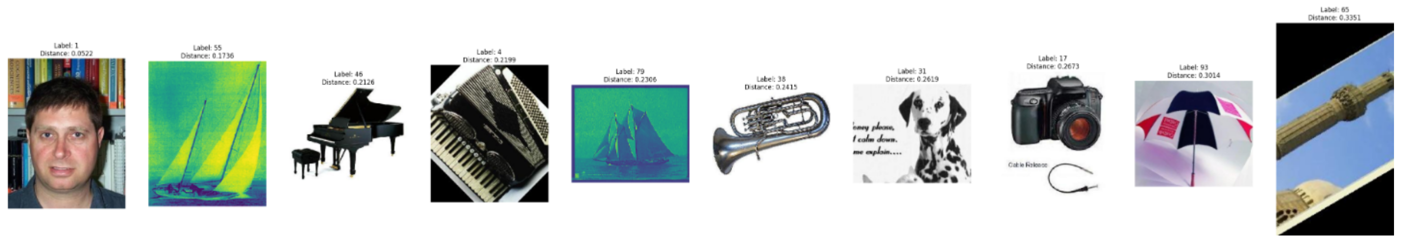
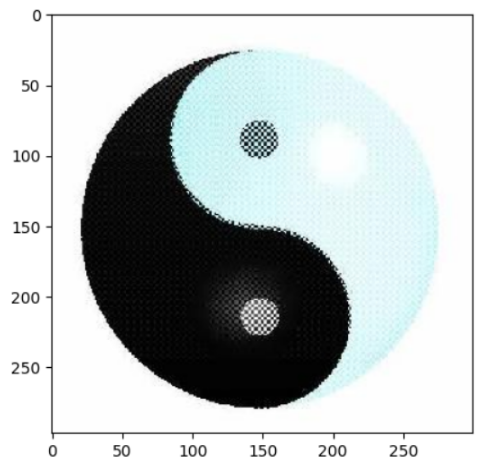
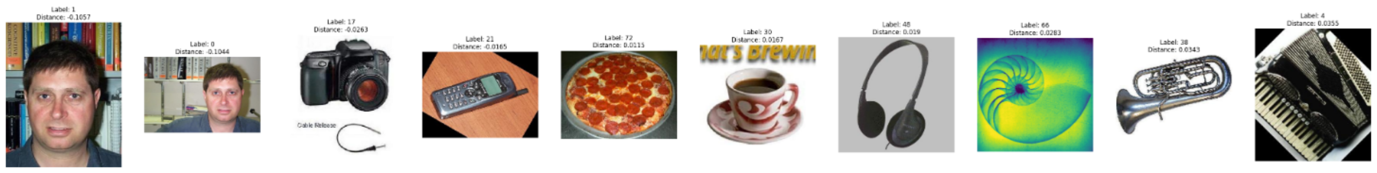


image {8676}; latent space {T5-avgpool-5-NNMF}; k=10





TASK 9

Specification:

User is required to input the latent semantic of the choice, and the specification for that semantic, namely feature model, K value, dimensionality reduction technique, followed by label number and then K2 value denoting the number of similar objects to find. The program will output the list of K2 most similar labels to the input label.

Description:

The program works by first calculating the appropriate label representative using the given label and latent semantic input. Then this label representative is compared to other vectors in the latent space using euclidean measure (or KL divergence is case of LDA dimensionality reduction), and the k2 most likely labels are shown to the user.

Design:

1. Inputs are obtained, and the latent semantics are loaded from the corresponding file.
2. For LS3

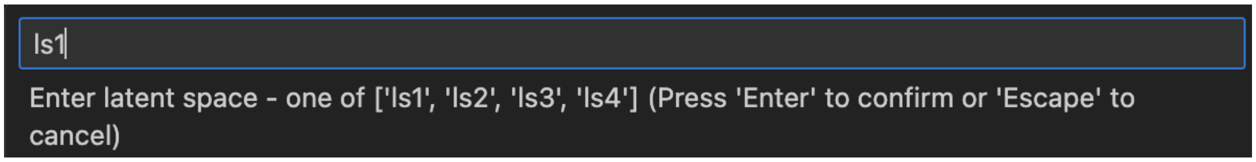
   a. For SVD, label-semantic is loaded from the saved latent semantics file. A comparison_feature_space is calculated by multiplying this label-semantic matrix with the semantics-core matrix.
   b. For other dimensionality reduction methods, the image-semantic matrix is taken as the comparison_feature_space
   c. A row corresponding to the input label is picked from the comparison_feature_space, which will be used as the comparison_vector
   d. Euclidean distance is calculated between this comparison_vector and other rows of the comparison_feature_space.
   e. Top K2 labels are shown as the output, signifying K2 most similar labels to the input label

3. For LS1 and LS4:
   a. A label_rep i.e. a representative for the label, is calculated by taking the mean of all the image feature descriptors from the database belonging to the same label
   b. For SVD, U and S matrices are multiplied to obtain comparison_feature_space. And label_rep vector is multiplied with V and S matrices to obtain comparison_vector. For LS4 however, label_rep is calculated using the rows of the U matrix. And comparison_vector is the multiplication of the label_rep and S.
   c. For NMF, comparison_vector is obtained by running nmf on the label_rep vector. W is used as the comparison_feature_space. For LS4, we calculate comparison_vector by calculator label_rep from within the H matrix.
   d. For k means, S matrix contains the coordinates of the K centroids. We calculate K dimensional comparison_vector by finding the euclidean distances between the label_rep and each centroid. For LS4, we just use the label_rep which will be calculated from the image-semantic matrix itself. Image-semantic matrix is used as comparison_feature_space.
   e. For LDA, for LS4, label_rep is calculated from the image-semantic matrix and used as comparison_vector. For LS1, the LDA model is run on the label_rep calculated from the database. Image-semantic matrix is used as a comparison_feature_space.
   f. Euclidean distances are calculated between this comparison_vector and every other row of the comparison_feature_space.
   g. K2 most similar images, each having unique labels are shown as the output.

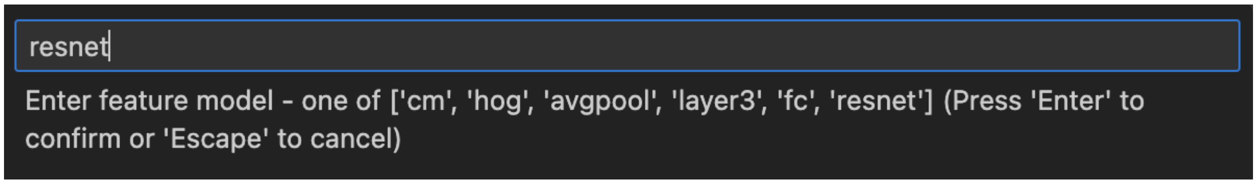
4. For LS2
   a. Label-semantic matrix and semantics-core matrices are loaded from the saved latent semantic file
   b. Multiplication of label-semantic and semantics-core matrix gives the comparison_feature_space
   c. A row is picked from this matrix corresponding to the input label to be used as the comparison_vector
   d. Euclidean distances are calculated between this comparison_vector and every other row of the comparison_feature_space.

e. Top K2 labels are shown as the output, signifying K2 most similar labels to the input label

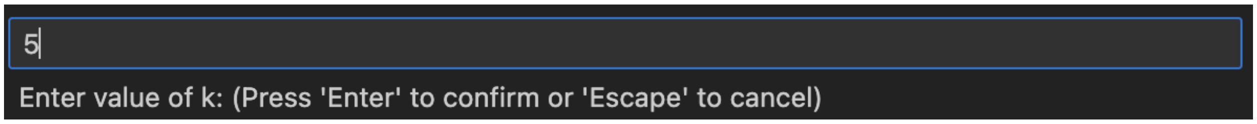
Input:



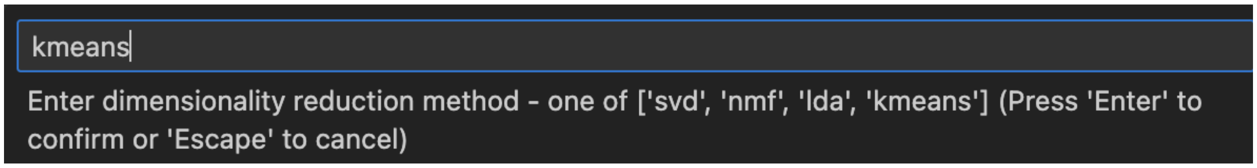
```
ls1
Enter latent space - one of ['ls1', 'ls2', 'ls3', 'ls4'] (Press 'Enter' to confirm or 'Escape' to cancel)
```



```
resnet
Enter feature model - one of ['cm', 'hog', 'avgpool', 'layer3', 'fc', 'resnet'] (Press 'Enter' to confirm or 'Escape' to cancel)
```



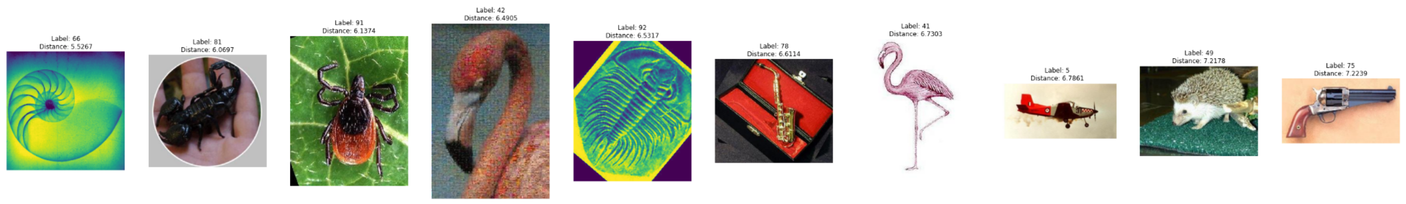
```
5
Enter value of k: (Press 'Enter' to confirm or 'Escape' to cancel)
```



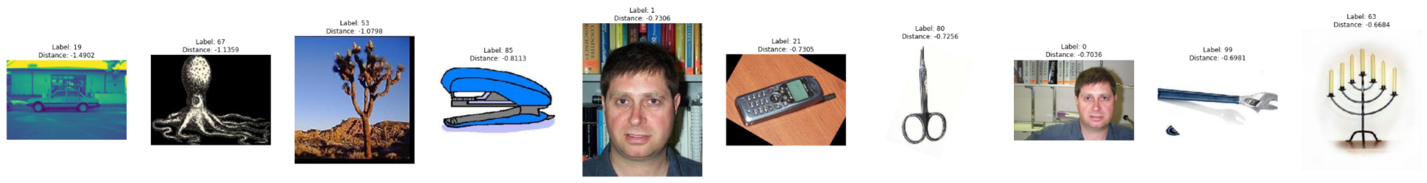
```
kmeans
Enter dimensionality reduction method - one of ['svd', 'nmf', 'lda', 'kmeans'] (Press 'Enter' to confirm or 'Escape' to cancel)
```

Output:

query label {100}; latent space {T3-RESNET-5-kmeans}; k=10



query label {20}; alt. latent space {T5-RESNET-5-kmeans}; k=10



TASK 10

Specification:

User is required to input the latent semantic of the choice, and the specification for that semantic, namely feature model, K value, dimensionality reduction technique, followed by label number and then K2 value denoting the number of similar objects to find. The program will output the list of K2 most similar images to the input label.

Description:

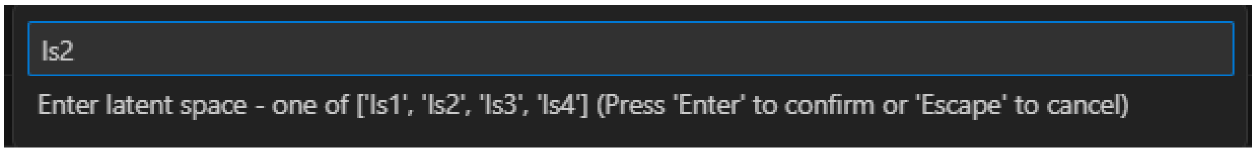
The program works by first calculating the appropriate label representative using the given label and latent semantic input. Then this label representative is compared to other vectors in the latent space using euclidean measure (or KL divergence is case of LDA dimensionality reduction), and the k2 most likely images are shown to the user.

Design:

1. The latent semantics data file is loaded
2. For LS3
    a. For SVD, image-semantic is loaded from the saved latent semantics file. A comparison_feature_space is calculated by multiplying this image-semantic matrix with the semantics-core matrix.
    b. For other dimensionality reduction methods, the image-semantic matrix is taken as the comparison_feature_space
    c. A row corresponding to the input label is picked from the comparison_feature_space, which will be used as the comparison_vector
    d. Euclidean distance is calculated between this comparison_vector and other rows of the comparison_feature_space.
    e. Top K2 images are shown as the output, signifying K2 most similar images to the input label
3. For LS1 and LS4:
    a. A label_rep i.e. a representative for the label, is calculated by taking the mean of all the image feature descriptors from the database belonging to the same label
    b. For SVD, U and S matrices are multiplied to obtain comparison_feature_space. And label_rep vector is multiplied with V and S matrices to obtain comparison_vector. For LS4 however, label_rep is calculated using the rows of the U matrix. And comparison_vector is the multiplication of the label_rep and S.
    c. For NMF, comparison_vector is obtained by running nmf on the label_rep vector. W is used as the comparison_feature_space. For LS4, we calculate comparison_vector by calculator label_rep from within the H matrix.
    d. For k means, the S matrix contains the coordinates of the K centroids. We calculate K dimensional comparison_vector by finding the euclidean distances between the label_rep and each centroid. For LS4, we just use the label_rep which will be calculated from the image-semantic matrix itself. Image-semantic matrix is used as comparison_feature_space.

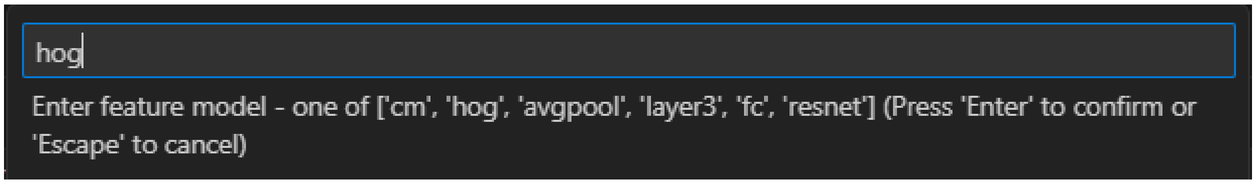
   e. For LDA, for LS4, label_rep is calculated from the image-semantic matrix and used as comparison_vector. For LS1, the LDA model is run on the label_rep calculated from the database. Image-semantic matrix is used as a comparison_feature_space.
   f. Euclidean distances are calculated between this comparison_vector and every other row of the comparison_feature_space.
   g. K2 most similar images are shown as the output.
4. For LS2
   a. Feature-semantic, image_semantic matrix and semantics_core matrices are loaded from the saved latent semantic file
   b. Multiplication of image_semantic and semantics-core matrix gives the comparison_image_space
   c. Multiplication of feature_semantic and label_rep gives comparison_feature_space which is then multiplied with semantics_core matrix to give the comparison_vector.
   d. Euclidean distances are calculated between this comparison_vector and every other row of the comparison_feature_space.
   e. Top K2 images are shown as the output, signifying K2 most similar images to the input label

Input:



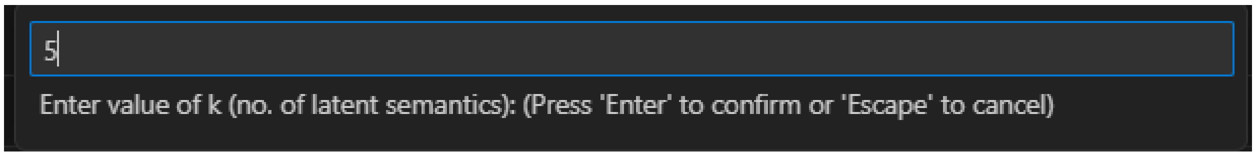
ls2

Enter latent space - one of ['ls1', 'ls2', 'ls3', 'ls4'] (Press 'Enter' to confirm or 'Escape' to cancel)



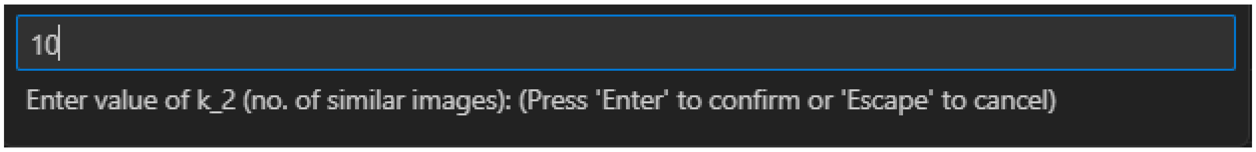
hog

Enter feature model - one of ['cm', 'hog', 'avgpool', 'layer3', 'fc', 'resnet'] (Press 'Enter' to confirm or 'Escape' to cancel)



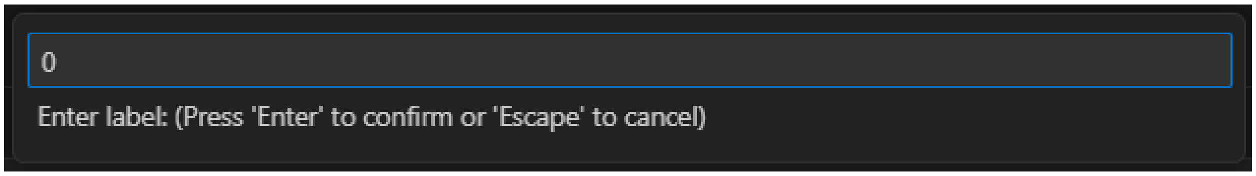
5

Enter value of k (no. of latent semantics): (Press 'Enter' to confirm or 'Escape' to cancel)



10

Enter value of k_2 (no. of similar images): (Press 'Enter' to confirm or 'Escape' to cancel)

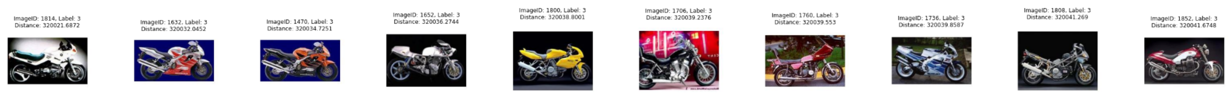


0

Enter label: (Press 'Enter' to confirm or 'Escape' to cancel)

Output:

query label {100}; alt. latent space {T4-HOG-5}; k=10



TASK 11

Specification:

User first specifies a feature model or latent space to construct an image-image similarity graph, along with parameters such as no. of similar images for each image. Then the user gives a specific label to apply the personalized pagerank algorithm and provides the number of most significant images, relative to that label, to identify from the graph.

Description:

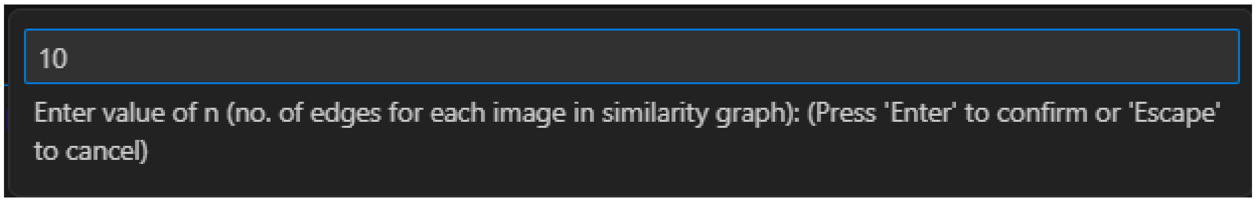
First, the similarity matrix for the given feature model is constructed using an appropriate distance measure, or loaded from the data files generated in T5 or T6 if it already exists. If latent space is given, it is constructed in a similar manner from the latent semantic matrices, using KL divergence for LDA space or euclidean distance for others. Using this similarity matrix, a similarity graph is constructed with the required n edges for each node, and the personalized PageRank (PPR) algorithm[2] is applied to find m most significant images similar to the given label.

Design:

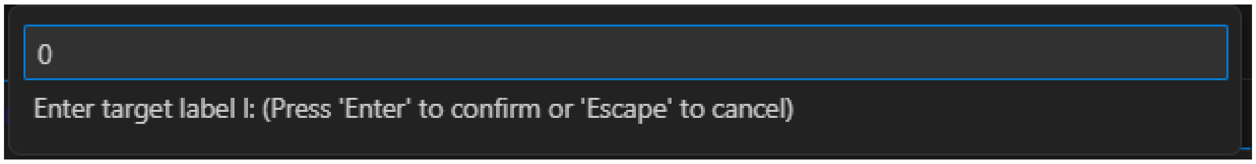
1. Inputs are obtained from the user, and the corresponding latent semantic data files are loaded.
2. Construction of similarity graph:
    a. Feature space:
       The image-image similarity matrix is loaded from T6 data files if the corresponding data exists, else it is calculated as in T6.
    b. Latent space:
       The image-image similarity matrix is calculated by multiplying the image-semantic and semantics-core matrices and then calculating the image-image similarity matrix on this latent space.
    c. Then, the similarity graph is constructed by taking n most similar images and forming edges as tuples of their image IDs.
3. Personalized PageRank:
    a. The PageRank algorithm, as described in its original implementation [2], is personalized by using the given label, and adding the teleportation probability only for nodes that are in the given label, scaled accordingly.
    b. The iterative approach is the same otherwise.

c. Convergence tolerance and other parameters are fine-tuned as experimentally needed.

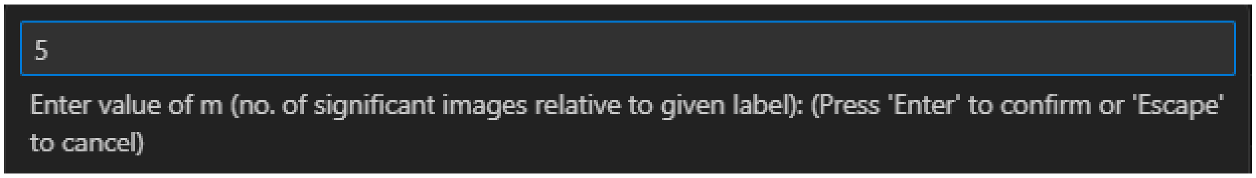
Input:



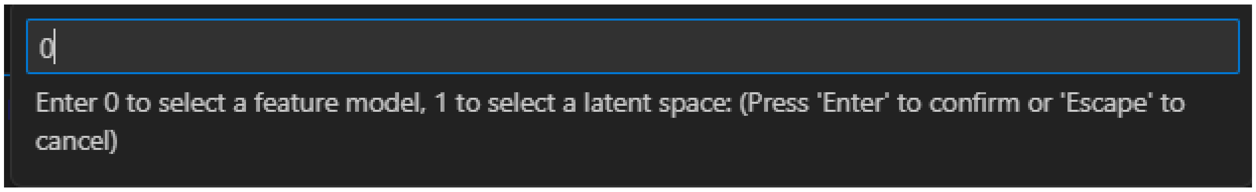
```
10
Enter value of n (no. of edges for each image in similarity graph): (Press 'Enter' to confirm or 'Escape' to cancel)
```



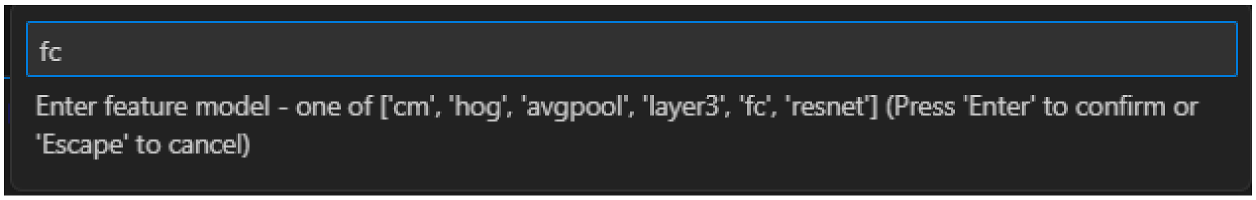
```
0
Enter target label l: (Press 'Enter' to confirm or 'Escape' to cancel)
```



```
5
Enter value of m (no. of significant images relative to given label): (Press 'Enter' to confirm or 'Escape' to cancel)
```



```
0
Enter 0 to select a feature model, 1 to select a latent space: (Press 'Enter' to confirm or 'Escape' to cancel)
```

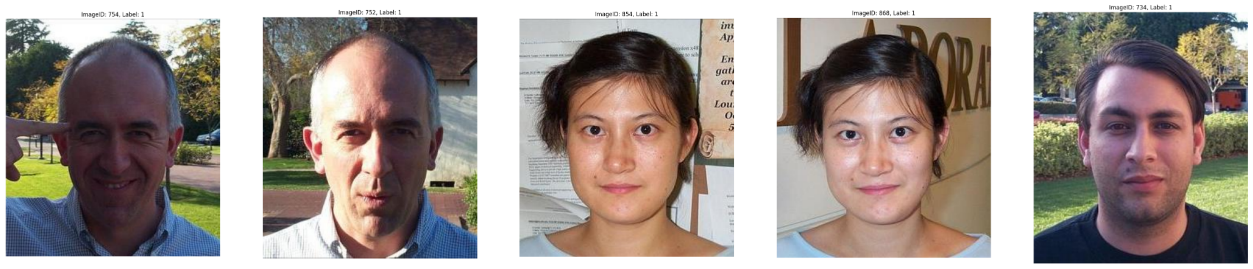


```
fc
Enter feature model - one of ['cm', 'hog', 'avgpool', 'layer3', 'fc', 'resnet'] (Press 'Enter' to confirm or 'Escape' to cancel)
```

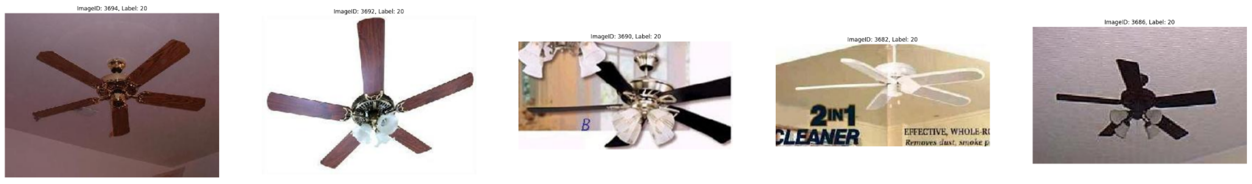
Output:

alt. space FC, n=10, m=5

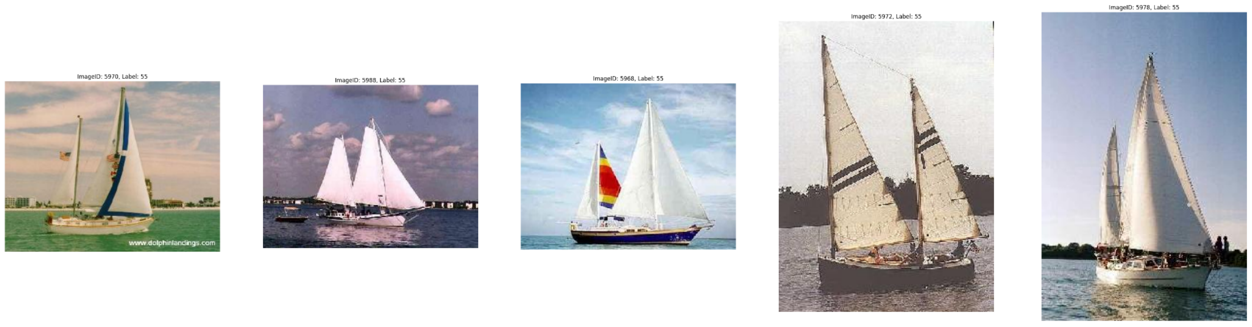
Query label = 0 (faces) - 287 iterations to convergence



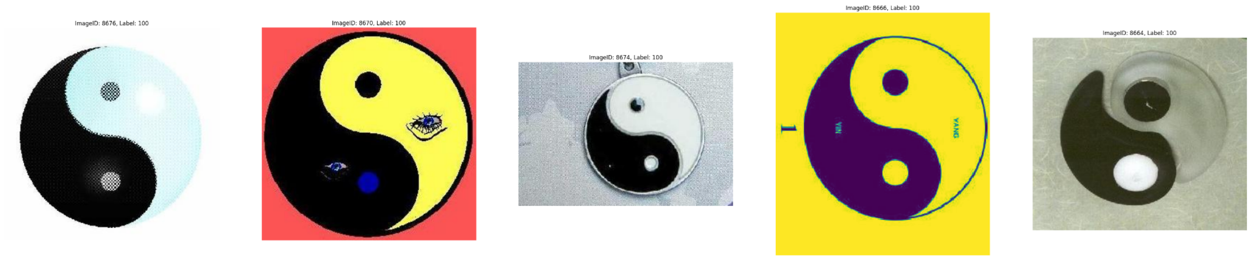
Query label = 20 (ceiling_fan) - 845 iterations to convergence



Query label = 55 (schooner) - 207 iterations to convergence



Query label = 100 (yin_yang) - 155 iterations to convergence


ImageID: 8676, Label: 100
ImageID: 8670, Label: 100
ImageID: 8674, Label: 100
ImageID: 8666, Label: 100
ImageID: 8664, Label: 100

INTERFACE SPECIFICATIONS

The runtime is the Jupyter notebook environment, allowing for step-through execution, cell by cell. It is also easily interactive in displaying images and getting user inputs. Matplotlib library is used to visualize images.

Feature Extraction and Image Retrieval (Task 0):

Users interact with a program to specify imageIDs, feature spaces, and the number of most similar images (k) they want to retrieve. The program provides a visualization of the most similar images with their scores under the selected feature space.

Label-Based Image Retrieval (Task 1):

Users provide a query label, a selected feature space, and the desired number of relevant images (k). The program retrieves and visualizes the most relevant images for the given label with their scores under the selected feature space.

Label Prediction (Task 2):

Task 2a: Users provide a feature space of their preference, a query imageID or image file, and the number of likely matching labels (k). With their scores, the program predicts and lists k most likely matching labels with their scores .
Task 2b: Users provide a query imageID or file and the number of likely matching labels (k). The program identifies and lists k most likely matching labels with their scores using the RESNET50 neural network model.

Latent Semantic Extraction (Tasks 3, 4, 5, 6):

Users select a feature model and enter the value of k. They also choose one of the following four dimensionality reduction techniques (SVD, NNMF, LDA, k-means) or CP decomposition. The program extracts latent semantics under the selected feature space, saves them to output files, and lists imageID-weight pairs ordered by weights for each latent semantic.

Image and Label Retrieval Using Latent Semantics (Tasks 7, 8, 9, 10):

Users provide an image file or an imageID, choose a latent space, and specify the number of similar images, matching labels, or relevant images (k). The program identifies, visualizes, or lists the desired results based on the selected latent space.

Similarity Graph Construction and Personalized PageRank (Task 11):

Users specify a feature model or latent space, values n and m, and a label l. The program constructs a similarity graph and utilizes personalized PageRank to determine the most significant images relative to the given label.

SYSTEM REQUIREMENTS/ INSTALLATION AND EXECUTION INSTRUCTIONS

Operating System - Agnostic

Hardware Requirements (Recommended):
16GB RAM, 4GB GPU memory, sufficient disk space (5+ GB)

Software Dependencies:
Python 3.7 or above.

Configurations:
Configuration of dataset path is needed as per host machine's directory system

Library Dependencies:
Necessary Python modules are listed in the requirements.txt file

Installations:
- MongoDB installation is required, either local or hosted/Atlas. Connection string may have to be modified as needed if not local.
- Users are required to install the Python libraries mentioned above.

Execution Instructions:
Jupyter Notebooks are preferred to run the code. To execute the tasks, the user can run the respective .ipynb files created for each task.

Miscellaneous:
The code is developed keeping a GPU-available system in mind. Minor modifications would be needed in case of execution on a non-GPU system, for the ResNet50 feature extraction part.

RELATED WORK

In the field of multimedia and web databases there have been studies that have laid the groundwork for our project's goals. These studies have provided insights and methodologies for analyzing, retrieving and reducing the dimensions of images. One notable advancement in image feature extraction is the use of trained neural network models like RESNET50. These models have played a role in extracting image features. Utilizing trained deep learning models for feature extraction has been widely documented in recent literature as it enhances the accuracy and efficiency of analyzing multimedia data. Such models enable us to extract high level features from images by utilizing neural network activations that capture intricate patterns.

Moreover graph based analysis as explored in Task 11 is rooted in network science and information retrieval. The application of PageRank for recommendation and ranking within graphs has been extensively studied. This study underscores the significance of personalization, in graph based analysis aligning with our goal of identifying images based on specific labels.

Extensive research has delved into techniques, like Singular Value Decomposition (SVD) (R. Bro, 1997) Nonnegative Matrix Factorization (NNMF) (D.D. Lee & H.S. Seung 1999) Linear Discriminant Analysis (LDA) (S. Roweis & L.K. Saul 2000) and k means clustering within the scope of latent semantics and dimensionality reduction tasks (Tasks 3 4 5 6). These techniques are frequently used in various domains for uncovering latent patterns and semantics within data, offering a foundation for the project's objectives.

CONCLUSION

In this phase of the project, we commenced on an extensive exploration of feature space processing and similarity measures, focusing on image feature extraction, dimensionality reduction, graph analysis, and latent semantics extraction. We began by extracting image features, enabling us to represent images in various feature spaces. Label-based image retrieval and label prediction tasks utilized the extracted features to offer users with relevant images and label predictions. Latent semantics extraction, revealed the hidden patterns within our data, which improved our ability to understand and analyze the data efficiently. Using personalized PageRank as an example, Task 11's graph-based analysis allowed us to identify the most significant images relative to a specific label, thus enhancing personalized recommendations and content ranking. This phase of the project highlighted the significance of combining image analysis, dimensionality reduction, and graph-based methodologies.

We observe that there are losses in accuracy when using latent semantic spaces, which is inevitable if the intrinsic dimensionality of the data exceeds the latent space's dimensions. But we also observe that using latent spaces for such processing has much less storage requirements and is also significantly faster than using original feature spaces. Moreover, using latent spaces allows us to identify inherent patterns in data. Such a tradeoff must be considered for real-world applications on a case-by-case basis.
Our rudimentary implementation of the PageRank algorithm's personalized version demonstrates the flexibility of graph-based analysis. Not only can such an algorithm adapt to incoming data thus making it feasible for real-world online learning situations, it can also construct different outcomes based on the number of edges in the graph. Further work could rely on weighted graphs, handle edge-cases like loops, and increase reliance on domain knowledge to specialize such algorithms for individual cases.

BIBLIOGRAPHY

1. K. Selçuk Candan, Maria Luisa Sapino, *Data Management for Multimedia Retrieval*, Cambridge University Press (2010)
2. Matthew D. Hoffman, David M. Blei, Francis Bach, *Online Learning for Latent Dirichlet Allocation*, NIPS'10: Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1, December 2010, p. 856-864
3. Sergey Brin, Lawrence Page, *The anatomy of a large-scale hypertextual Web search engine*, Computer Networks and ISDN Systems, Volume 30, Issues 1–7, (1998), p. 107-117

APPENDIX

In this phase, collaboration was achieved by playing to our individual strengths and covering our weaknesses. Kaushik, Niraj, Pranav handled most of the coding part, while Mohan, Pavan and Madhura handled most of the report part. We all contributed to both parts though.