Image Feature Extraction, Dimensionality Reduction & Optimized Latent Semantic Search: Multimedia & Web Databases

Khadyothan Choudari Dasari *Chandra Kishore Reddy Gurram* *Rohit Bathi*
Rahul Rajaram *Bevin Biju Thomas* *Lalit Arvind Balaji*

Abstract:

In the previous project phase, our primary focus was on utilizing five distinct features for image analysis: color moments, histogram of gradients, ResNet50 pre-trained neural architecture's average pool layer, layer3, and the fully connected layer. These features were employed to query the most similar k images from the database when provided with a query image and the value of k. However, a significant challenge emerged due to the high dimensions of these feature models (900, 1000, and 1024), resulting in data points clustering in the outer layer. This clustering posed problems for accurate querying using distance metrics and increased the risk of overfitting. To address these issues, we recognized the need for dimensionality reduction techniques to enhance search efficiency and mitigate overfitting. In this project phase, we explored various dimensionality reduction methods, including Singular Value Decomposition (SVD), Non-Negative Matrix Factorization (NMF), Latent Dirichlet Allocation (LDA), K-means, and CP Decomposition (Candecomp/Parafac). The choice of dimensionality reduction technique depended on the specific application and incorporated domain knowledge. For example, if our goal was to preserve the variance of the data, we opted for Principal Component Analysis (PCA). Alternatively, when our focus was on storage efficiency for the database, we chose SVD, which is a variant of PCA. We also considered the use of graphs to represent images, where edges represented similarities between images, allowing us to perform latent searches using the Personalized Page Rank algorithm.

Keywords:

Feature models, latent semantics, dimensionality curse, singular valued decomposition, non-negative matrix factorization, latent Dirichlet allocation, K-means, CP decomposition, page rank algorithm.

Introduction:

Key Terminology:

Feature Models: Feature models are representations of specific characteristics or attributes of data. In image analysis, they can describe aspects like color, texture, or shape, which are essential for understanding and processing visual information.
Latent Semantics: Latent semantics refer to hidden or underlying meanings within a dataset, often extracted through techniques like topic modelling or dimensionality reduction. This concept is crucial for uncovering implicit relationships and patterns in data.
Dimensionality Curse: The dimensionality curse describes the challenge of dealing with high-dimensional data. As the number of features or dimensions in a dataset increases, the complexity of analysis and computation grows, often leading to decreased accuracy and increased resource requirements.
Singular Valued Decomposition(SVD): SVD is a matrix factorization technique that decomposes a matrix into three separate matrices to identify the underlying structure and relationships within the data. It is frequently used for dimensionality reduction and data compression.
Non-Negative Matrix Factorization(NMF): NMF is a dimensionality reduction method that decomposes a matrix into non-negative components, making it particularly suitable for analysing non-negative data like images, text, or gene expression.
Latent Dirichlet Allocation(LDA): LDA is a probabilistic model used in natural language processing and topic modelling. It discovers latent topics within a collection of documents, helping to reveal the underlying themes in textual data.

K-means: K-means is a clustering algorithm that partitions data points into K clusters based on their similarity. It is widely used for grouping data and finding patterns within it.
CP Decomposition: CP decomposition is a tensor factorization technique that breaks down multi-dimensional data into a sum of rank-one tensors. It's used in various applications, including image and signal processing.
Page Rank Algorithm: PageRank is an algorithm developed by Google to rank web pages in search engine results. It evaluates the importance of web pages by considering the number and quality of links pointing to them, making it a fundamental concept in web search and network analysis.

Project Goal/Objective:

The primary goal or objective of the project is to apply various dimensionality reduction techniques to the feature models used for image analysis. By doing so, the project aims to enhance search efficiency and reduce the risk of overfitting when querying a database of images. The project seeks to allow users to input an image or image label and, based on their specifications, visualize the k most similar images or labels by navigating through the latent semantics created through dimensionality reduction. In essence, the project aims to provide an efficient and effective means of exploring and retrieving relevant images or labels from a dataset using latent representations derived from these advanced techniques.

Assumptions:

Handling Grayscale Images: Grayscale images are assumed to be treated as three-channel images by replicating the single channel three times. This ensures consistency in processing when extracting feature descriptors from both grayscale and color images.
Image Feature Descriptors Normalization: It is assumed that the image feature descriptors are not normalized during the extraction process. The descriptors will retain their original scales and values, which may vary depending on the specific feature extraction techniques used.

Proposed Solution/Implementation:

Database Integration:

The project utilizes the MongoDB database for efficient storage and retrieval of image data. The PyMongo Python library is employed to establish a connection between the Python code and the MongoDB database, allowing for seamless data management. MongoDB employs collections to store data in a structured format known as documents, which are organized in a JSON-like structure.

Image Preprocessing:

User Input: The code is designed to accept user input in various forms, including image IDs, new image files not present in the database, or image labels. Regardless of the input type, the code processes it to create an image vector based on the provided information.
Grayscale Image Handling: The code includes logic to detect grayscale images. If an image is grayscale, it will be converted into a 3-channel representation to ensure consistent processing across all images and stored in the database collection.
Image Standardization: To ensure consistency, all images are resized to a uniform size of 300x100 pixels. This standardization simplifies subsequent feature extraction and ensures that images have consistent dimensions.

Database Storage:

The computed feature descriptors for each image will be saved in the MongoDB database, and they will be associated with their respective image IDs and image category labels. This proposed implementation

facilitates efficient feature extraction and the storage of image data, which, in turn, allows for subsequent analysis and retrieval of relevant visual features. The use of MongoDB and standardization techniques ensures scalability and consistency in managing a wide range of image datasets.

Furthermore, within our database, we have a collection of representative images for each label. When a specific label is provided as input, the database is utilized to retrieve the representative image vector associated with that input label. These representative images are generated by selecting the image closest to the mean vector calculated from all the images within that particular label category. This approach helps capture the essence of each category and provides a means for efficient retrieval of characteristic images.

Task 0: Implementation of Phase1

Query Input Given:

Given the Caltech101 dataset, we need to map even labelled images into 5 different feature spaces and store them in database. Now given a feature space and a k as input, we need to identify and visualize k similar images, along with the scores.

Implementation:

A program has been developed to extract feature descriptors, including color moments, histogram of gradients, ResNet50 neural network's layer3, average pool layer, and fully connected layer. This program's objective is to identify and display the k most similar images to a specified input image ID based on each of these visual models. The process entails selecting suitable distance or similarity measures for each feature model and then computing distance or similarity scores by comparing the input image's feature descriptors with those of all other images stored in the database.

Design Choices & Decisions:

The distance function used to compute the distance score of given images with the database image feature model vectors is Euclidean for all the feature models. The Euclidean distance is a commonly used distance metric for comparing feature vectors. It is appropriate for continuous and numeric feature descriptors like Color Moments, HOG, and ResNet50 features. It measures the straight-line distance between two points in a multidimensional space, which makes it suitable for assessing the similarity between feature vectors.

Expected Outcomes:

The top k similar images are displayed based on the distance scores with the query vector.

Outputs:

The output images are stored in Task0 directory as separate directories for each input image ID, including 0, 880, 2500, 5122, and 8676. Within each of these directories, all five feature descriptor images are stored. Certainly, below are two examples of the results for the top 10 visually similar images, including their distance scores, for Image ID 2500 (Color Moments) and Image ID 880 (ResNet50 Layer 3). The remaining outputs for all feature descriptors in the input images can be located in the "results" directory.

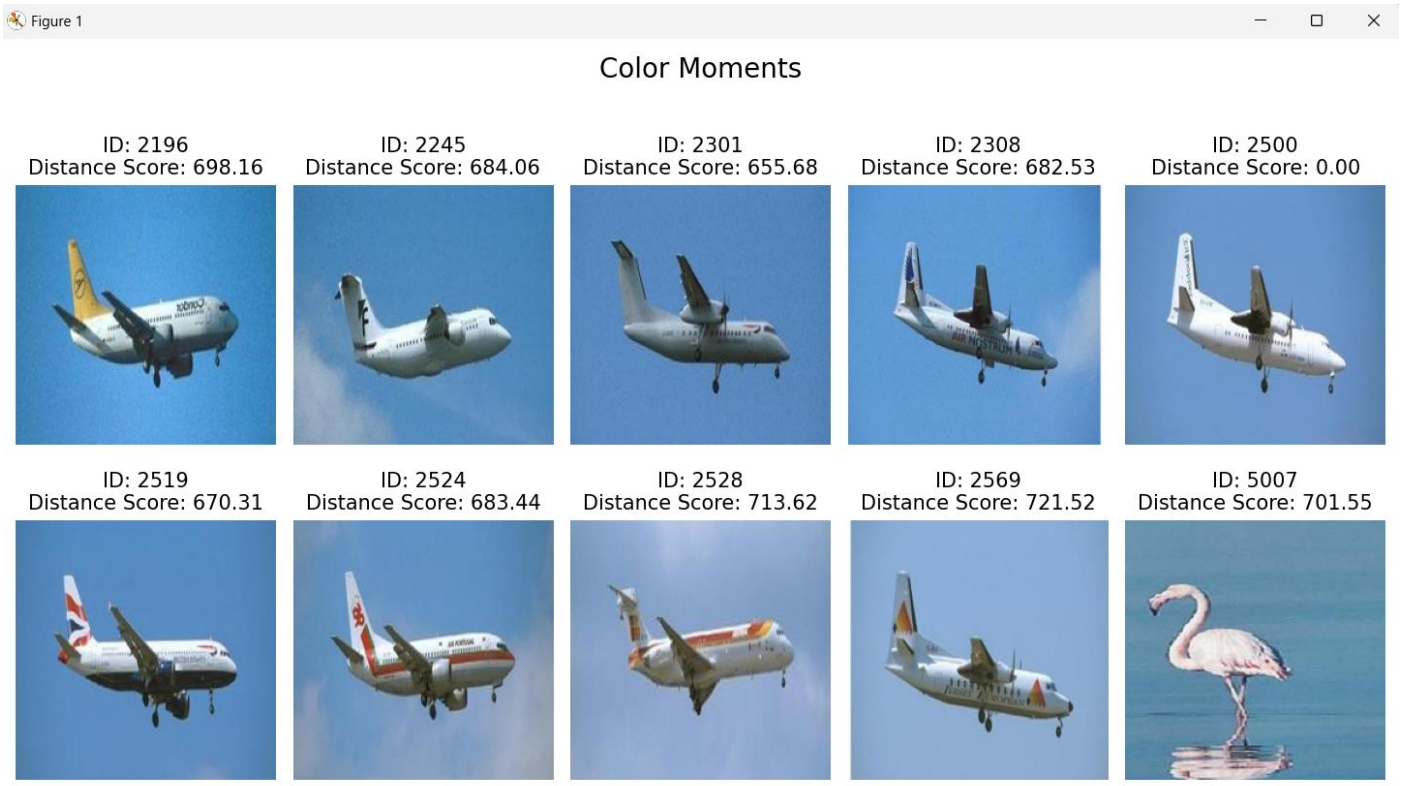

Figure 0.1: Color Moments Output for Image ID 2500

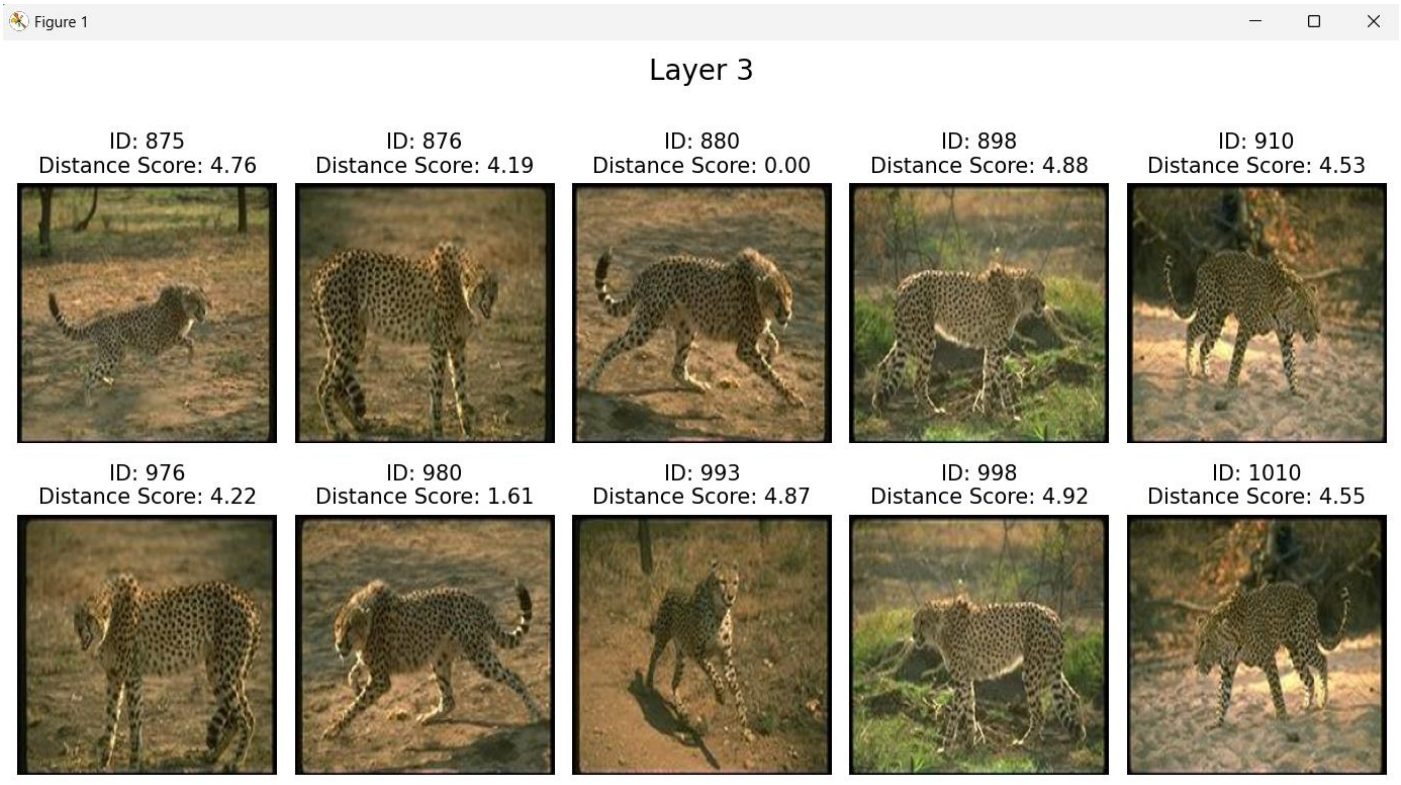

Figure 0.2: ResNet50 Layer 3 Output for Image ID 880

Assumptions & Observations:

we observed that the color moments feature extraction for Image ID 0 produced results that differed from our expectations. This discrepancy highlights a limitation of using the Euclidean distance metric for color

moments, as it may not be the most efficient measure for this specific feature model. However, it's worth noting that the rest of the images in our dataset yielded satisfactory results with color moments and for the image ID 5122, which was provided as a grayscale input, we applied a mapping technique to convert it into a 3-channel representation before extracting feature descriptors.

Task 1: Implementation of Task0 but Query Label as Input

Query Input Given:

Given a query label as input need to visualize and display the most relevant k images for the given label l, along with their scores.

Implementation:

The given input labels centroid is computed, which is the image closest(most similar) to the mean vector of all the images of the given label l. Now k-nearest images of the centroid are identified using distance metrics and are visualized.

Design Choices & Decisions:

The distance function used to compute the distance score of the centroid image of the given label with the database image feature model vectors is Euclidean for all the feature models. This distance metric is used for the same reason as in Task0.

Expected Outcomes:

The top k relevant images of the input query label are displayed along with the distance scores.

Outputs:

Similar to the outputs stored in Task0, all the outputs for the given input query label{0, 20, 55, 100} are stored in task1 directory in the outputs folder for all the 5 feature descriptor models. Certainly, below are two examples of the results for the top 10 visually similar images, including their distance scores, for Label 55 (Color Moments) and Label 100 (ResNet50 Layer FC).

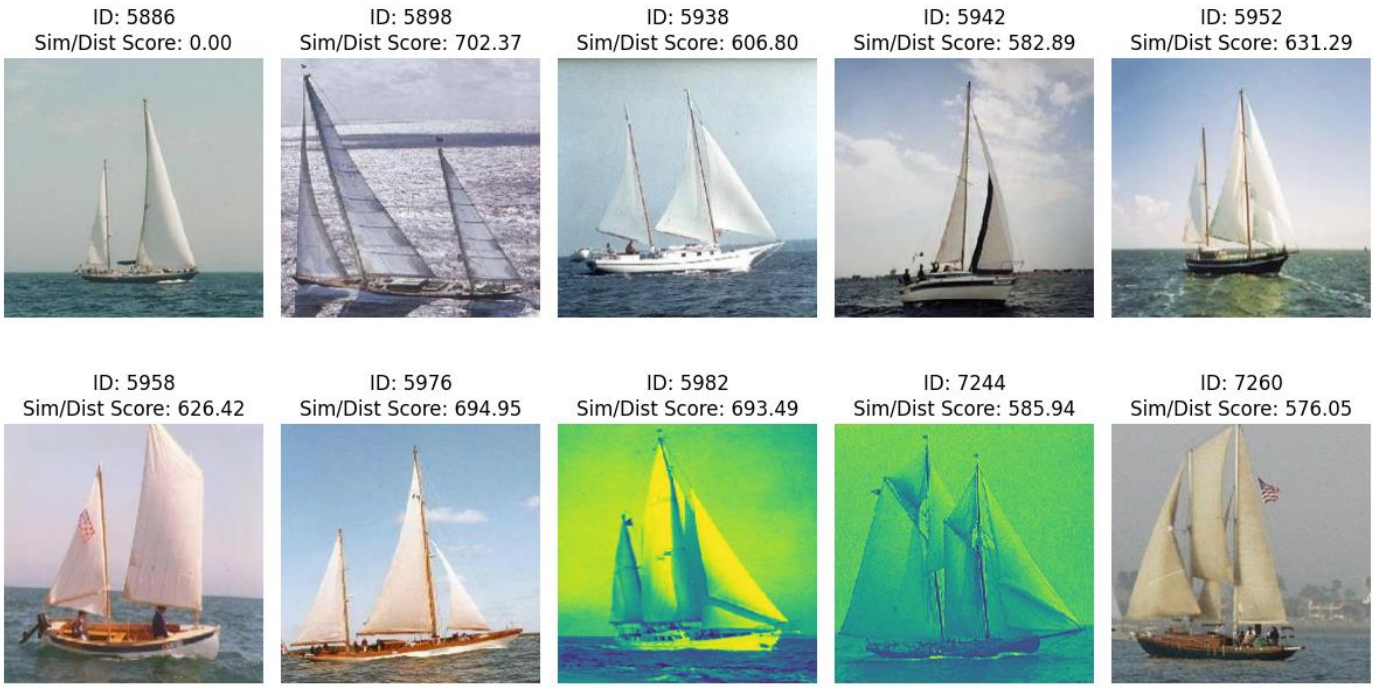

Figure 1.1: Color Moments output for Label 55

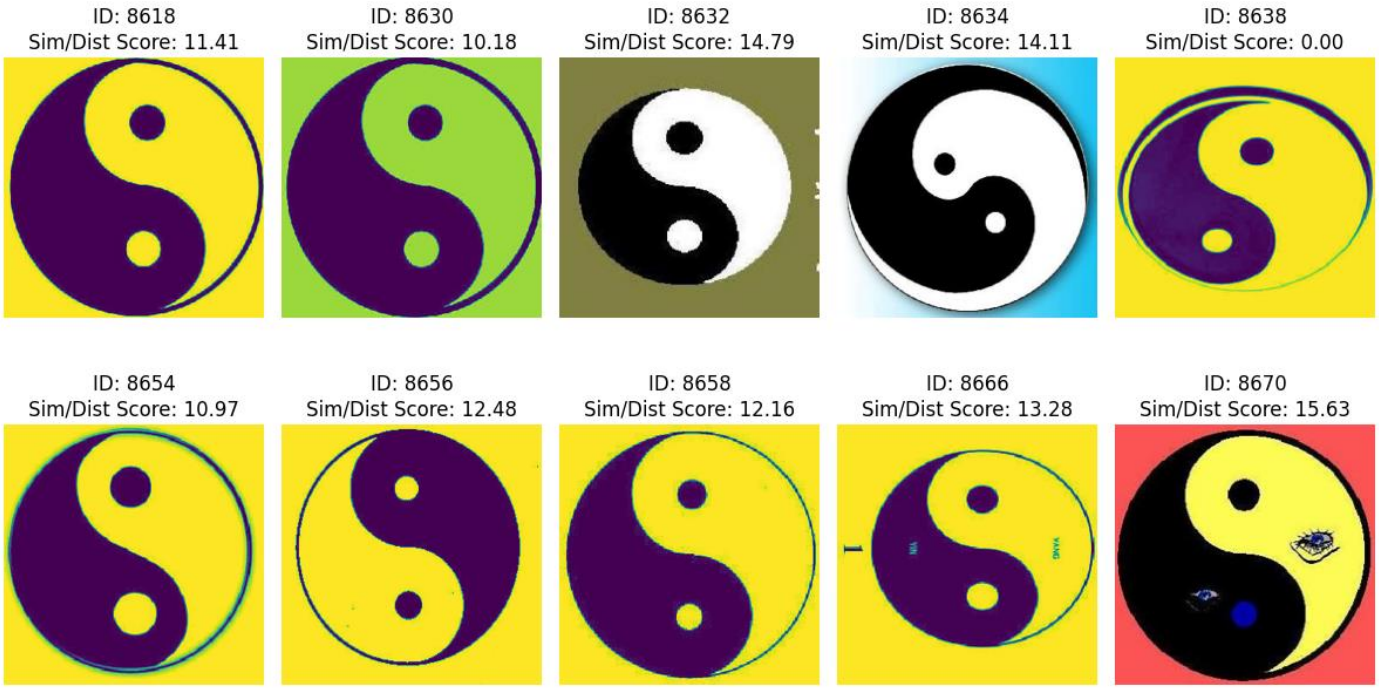

Figure 1.2: Resnet50 FC Layer Output for Label 100

Assumptions & Observations:

Upon analysing the results obtained from all the outputs, we noted that the Color Moments and Histogram of Gradients (HOG) feature descriptors are particularly accurate for labels associated with images primarily centred on a subject, featuring uniform color in the surroundings(Example label-5, label-55). In contrast, the ResNet features consistently provide accurate results across various label types. It is worth mentioning that not standardizing the Color Moments and HOG feature descriptors and applying distance functions leads to higher distance similarity scores, which may not be ideal for predicting all label types. To enhance the results, we should explore alternative methods, such as using cosine similarity and other approaches, to further refine the accuracy of our predictions.

Task 2: Identifying Distinct K-Similar Labels for a Given Input Image

Query Input Given:

Given a query image id and feature space as input need to visualize and display the most likely matching distinct k labels, along with their scores.

Implementation:

All the representative images of the labels are calculated and stored in a list. The input labels representative image is used and find the similarity of the query image vector with all the representative images of the label are calculated. The top k likely matching labels with high similarity are identified and listed.

Task2b:

Instead of attaching hooks at different layers of the resnet50 neural network, we run the full resnet50 on the image and get the final vector and consider this as the feature model. The final layer of the resnet50 has a SoftMax layer after the fully connected layer.

Design Choices & Decisions:

Cosine similarity is used to find the similarity scores between label representative vector. Since, we observed some results might not be accurate using the Euclidean distance in the previous tasks, cosine similarity is used in this task and observed how the results are obtained.

Expected Outcomes:

The top k distinct similar labels of the input query label are displayed along with the similarity scores.

Outputs:

Certainly, below are the results for the top 10 distinct similar labels including their similarity scores. First 6 figures are the outputs of task2a and the last image figure is the one of the outputs of task2b.

```
Enter input color_moments
Enter k: 10
Label: 0, Similarity: 0.946
Label: 13, Similarity: 0.942
Label: 9, Similarity: 0.940
Label: 39, Similarity: 0.940
Label: 66, Similarity: 0.939
Label: 41, Similarity: 0.939
Label: 51, Similarity: 0.939
Label: 23, Similarity: 0.938
Label: 1, Similarity: 0.938
Label: 63, Similarity: 0.938
```

Fig 2.1: Image ID 0, CM

```
Enter input hog
Enter k: 10
Label: 2, Similarity: 0.920
Label: 97, Similarity: 0.813
Label: 82, Similarity: 0.790
Label: 87, Similarity: 0.780
Label: 42, Similarity: 0.777
Label: 54, Similarity: 0.776
Label: 31, Similarity: 0.775
Label: 29, Similarity: 0.773
Label: 51, Similarity: 0.772
Label: 74, Similarity: 0.769
```

Fig 2.2: Image ID 880, HOG

```
Enter input resnet50_avgpool
Enter k: 10
Label: 5, Similarity: 0.826
Label: 50, Similarity: 0.487
Label: 51, Similarity: 0.269
Label: 71, Similarity: 0.265
Label: 33, Similarity: 0.258
Label: 62, Similarity: 0.253
Label: 14, Similarity: 0.248
Label: 74, Similarity: 0.246
Label: 65, Similarity: 0.244
Label: 6, Similarity: 0.235
```

Fig 2.3: Image ID 2500, AvgPool

```
Enter input resnet50_layer3
Enter k: 10
Label: 43, Similarity: 0.930
Label: 83, Similarity: 0.920
Label: 13, Similarity: 0.892
Label: 80, Similarity: 0.891
Label: 48, Similarity: 0.891
Label: 63, Similarity: 0.890
Label: 67, Similarity: 0.888
Label: 91, Similarity: 0.887
Label: 99, Similarity: 0.884
Label: 100, Similarity: 0.883
```

Fig 2.4: Image ID 5122, Layer3

```
Enter input resnet50_fc
Enter k: 10
Label: 100, Similarity: 0.602
Label: 48, Similarity: 0.479
Label: 84, Similarity: 0.448
Label: 85, Similarity: 0.342
Label: 17, Similarity: 0.342
Label: 21, Similarity: 0.295
Label: 45, Similarity: 0.251
Label: 80, Similarity: 0.239
Label: 99, Similarity: 0.223
Label: 13, Similarity: 0.216
```

Fig 2.5: Image ID 8676, FC

```
Enter input hog
Enter k: 10
Label: 76, Similarity: 0.853
Label: 44, Similarity: 0.840
Label: 28, Similarity: 0.826
Label: 37, Similarity: 0.824
Label: 58, Similarity: 0.822
Label: 54, Similarity: 0.810
Label: 29, Similarity: 0.805
Label: 86, Similarity: 0.804
Label: 10, Similarity: 0.800
Label: 47, Similarity: 0.800
```

Fig 2.6: Image ID 7122, HOG

```
Enter query image ID: 2500
Enter k: 10
Top 10 labels:

label id:0, category: Faces_2, distance: 4.716383363890912

label id:1, category: Faces_3, distance: 4.716383363890912

label id:2, category: Leopards, distance: 4.716383363890912

label id:3, category: Motorbikes_16, distance: 4.716383363890912

label id:4, category: accordion, distance: 4.716383363890912

label id:5, category: Airplanes_Side_2, distance: 4.716383363890912

label id:6, category: anchor, distance: 4.716383363890912

label id:7, category: ant, distance: 4.716383363890912

label id:8, category: barrel, distance: 4.716383363890912

label id:9, category: bass, distance: 4.716383363890912
```

Fig 2.7: Image ID 2500, Task2b (SoftMax Resnet)

Assumptions & Observations:

From the outputs seen it can be noted that for each image input the most similar label is the actual label it belongs to and the following labels are the similar labels along with their similarity scores. The cosine similarity metric used gave accurate results in predicting the top k distinct similar labels.

Task 3: Dimensionality Reduction (SVD, NNMF, LDS, K-MEANS)

Query Input Given:
User gives the feature model, k and a dimensionality technique. Need to store the top-k latent semantics after performing the query dimensionality reduction technique on the given feature model and list the image-id weight pairs in descending order.

Implementation:
The given input feature models Data Matrix is created with the images as rows and columns as the image's dimensions. This will be a (4339x900) matrix for HOG or color moments. This matrix is then sent to the respective dimension reduction technique based on the user's choice. The dimension reduction technique returns the top-k latent semantics for the feature model which is in this case (4339xK) matrix.

This latent semantics are stored in Json files with the naming convention of `*{dim_red_tech}_{feature_model}_{k}_ls.json*`. The latent semantics are used to calculate the image-id weight pairs. Image-ID weight pairs are calculated by taking the L2 norm of the latent space vector of each image and stored in a txt file with naming convention `*{dim_red_tech}_{feature_model}_{k}.txt*`. All these files are stored in folders called latent semantics and weight pairs in a directory called outputs in the code directory. We created the 4339 * 900 (for Color_moments, HOG) and (4339 * 1024 or 4339 * 1000 for ResNet) data matrix by running a script and stored alongside the code.

SVD:

a. Design Choices & Decisions: SVD is implemented manually without using any libraries. The covariance matrix of the data is calculated. Using the covariance matrix, the eigen vectors and values are calculated. Eigen vectors with highest eigen values signifies highest variance preserved in that direction. These vectors are not considered as the basis for the latent semantics. The decomposition gives 3 matrices U, S & VT. The dot product of U & S gives the latent semantics.

b. Expected Outcomes: The expected outcome of this code is to perform dimensionality reduction using SVD and generate a list of image ID-weight pairs based on the latent semantics. The U reduced, S reduced, and VT reduced matrices are intermediate results, and latent semantics is the product of these matrices.

c. Outputs:

```
dasar@Khadyu MINGW64 /c/Khadyu/ASU/Fall 2023/Multimedia & Web Databases/Project/phase2_project/code/tasks
$ python task3.py

Select a feature model(Select one among these):
1. color_moments
2. hog
3. resnet50_layer3
4. resnet50_avgpool
5. resnet50_fc
6. resnet50_softmax

Enter input: color_moments
Enter k value: 5
Enter one of the following dimensional reduction techniques on the chosen feature model:

1. SVD
2. NNMF
3. LDA
4. k-means

Enter your choice: 1
(4339, 900)
(4339, 5)
```

Fig 3.1: SVD on Color Moments, k=5

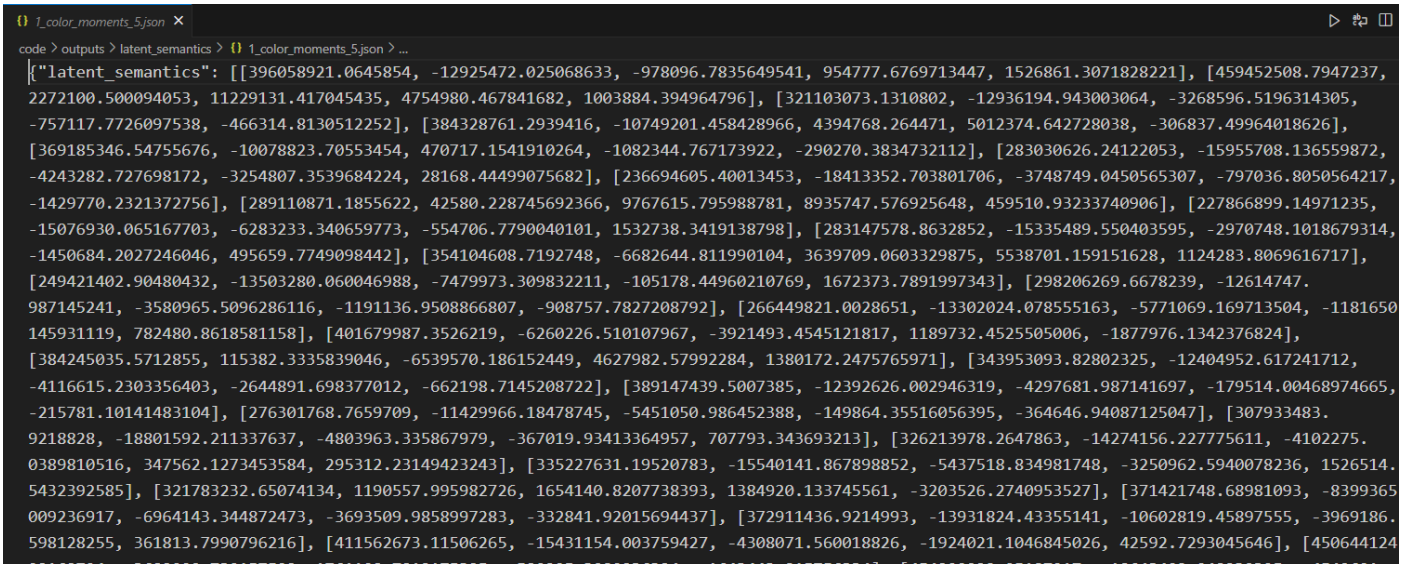

Fig 3.2: Latent Semantics of Color Moments, k=5

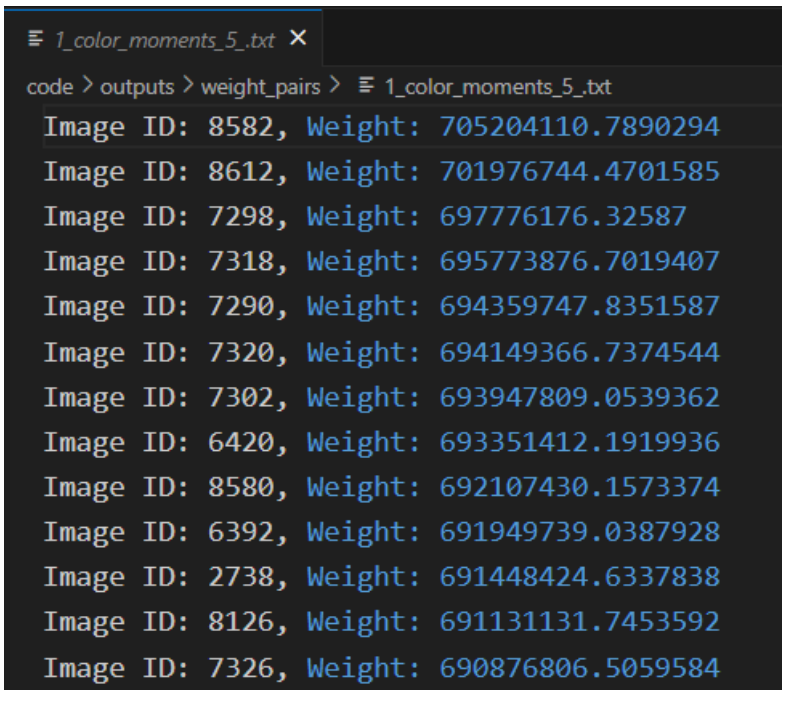

Fig 3.3: Image-ID Weight pairs of CM, k=5

d. Assumptions & Observations & Improvements: L2 norm of reduced features of the image are assumed to be the weight of the image.

NNMF:

a. Design Choices & Decisions: NNMF is implemented manually without using any external libraries. NNMF enforces non-negativity constraints on the factorization matrices. The code presumably ensures that all matrices involved in the factorization process remain non-negative, which is a crucial characteristic of NNMF.

b. Expected Outcomes: The expected outcome of this code is to perform NNMF on a given data matrix, factorizing it into two non-negative matrices (W and H) such that the product approximates the original data matrix. The matrices W and H represent the latent semantics or basis vectors that capture underlying patterns in the data.

c. Outputs:

```
dasar@Khadyu MINGW64 /c/Khadyu/ASU/Fall 2023/Multimedia & Web Databases/Project/phase2_project/code/tasks
$ python task3.py

Select a feature model(Select one among these):
1. color_moments
2. hog
3. resnet50_layer3
4. resnet50_avgpool
5. resnet50_fc
6. resnet50_softmax

Enter input: resnet50_softmax
Enter k value: 5
Enter one of the following dimensional reduction techniques on the chosen feature model:

1. SVD
2. NNMF
3. LDA
4. k-means

Enter your choice: 2
(4339, 1000)
```

Fig 3.4: NNMF on Resnet, k=5

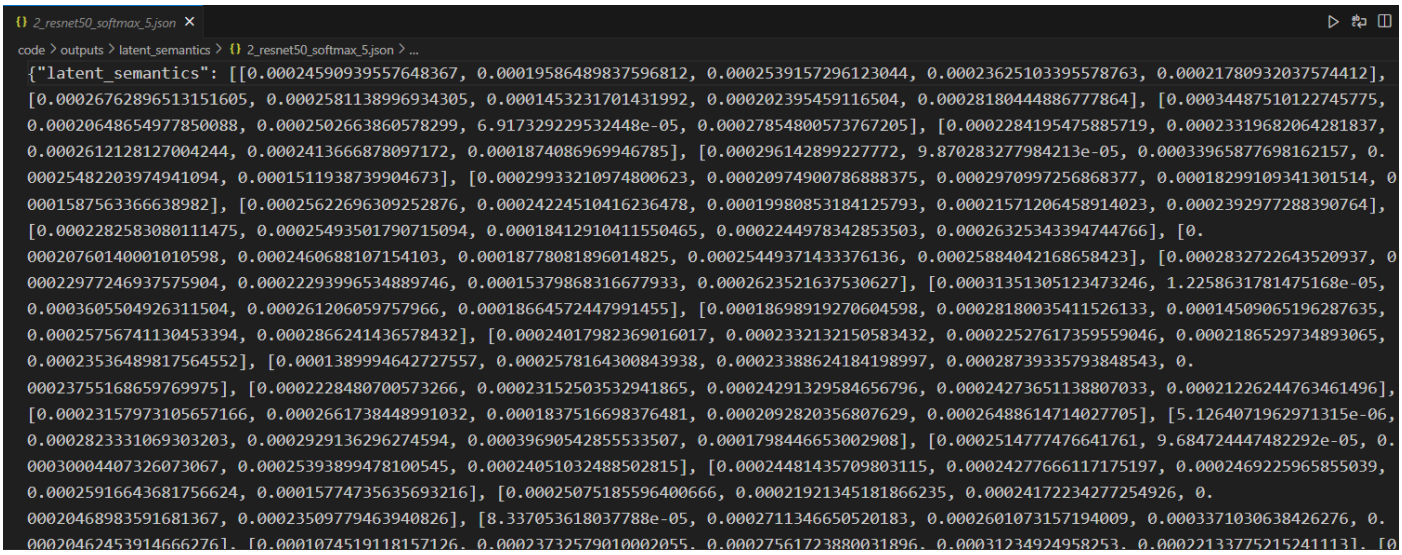

Fig 3.5: Latent Semantics of Resnet, k=5

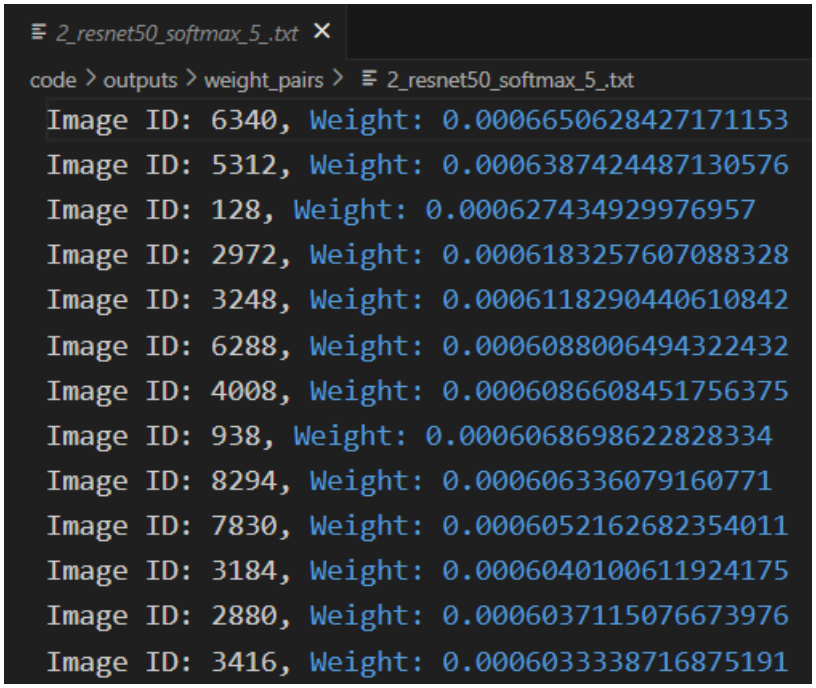

Fig 3.6: Image-ID weight pairs of Resnet, k=5

d. Assumptions & Observations & Improvements: L2 norm of reduced features of the image are assumed to be the weight of the image.

LDA:

a. Design Choices & Decisions: Sk-learn LDA model is chosen instead of Gensim library's LDA model. We take the minimum value of the data matrix, if it is less than 0 then we add the minimum value to all the values in the data matrix because the model takes only positive values as input

b. Expected Outcomes: The reduced features of the data matrix of selected feature descriptor

c. Outputs:

```
lalitarvind@Lalits-MacBook-Pro tasks % python3 task3.py

Select a feature model(Select one among these):
1. color_moments
2. hog
3. resnet50_layer3
4. resnet50_avgpool
5. resnet50_fc
6. resnet_softmax


Enter input: color_moments
Enter k value: 5
Enter one of the following dimensional reduction techniques on the chosen feature model:

1. SVD
2. NNMF
3. LDA
4. k-means

Enter your choice: 3
iteration: 1 of max_iter: 10
iteration: 2 of max_iter: 10
iteration: 3 of max_iter: 10
iteration: 4 of max_iter: 10
iteration: 5 of max_iter: 10
iteration: 6 of max_iter: 10
iteration: 7 of max_iter: 10
iteration: 8 of max_iter: 10
iteration: 9 of max_iter: 10
iteration: 10 of max_iter: 10
[[0.21007992 0.14297401 0.18229707 0.17875757 0.28589144]
 [0.3764684  0.09038292 0.1665268  0.11070077 0.25592112]
 [0.13975488 0.1954268  0.18526866 0.25754409 0.22200557]
 ...
 [0.30264842 0.22017843 0.118091   0.17345674 0.18562541]
 [0.49766307 0.10331386 0.11079004 0.07919205 0.20904097]
 [0.59399551 0.12471663 0.08625037 0.06089582 0.13414167]]
```

Fig 3.1: LDA on Color Moments, k=5

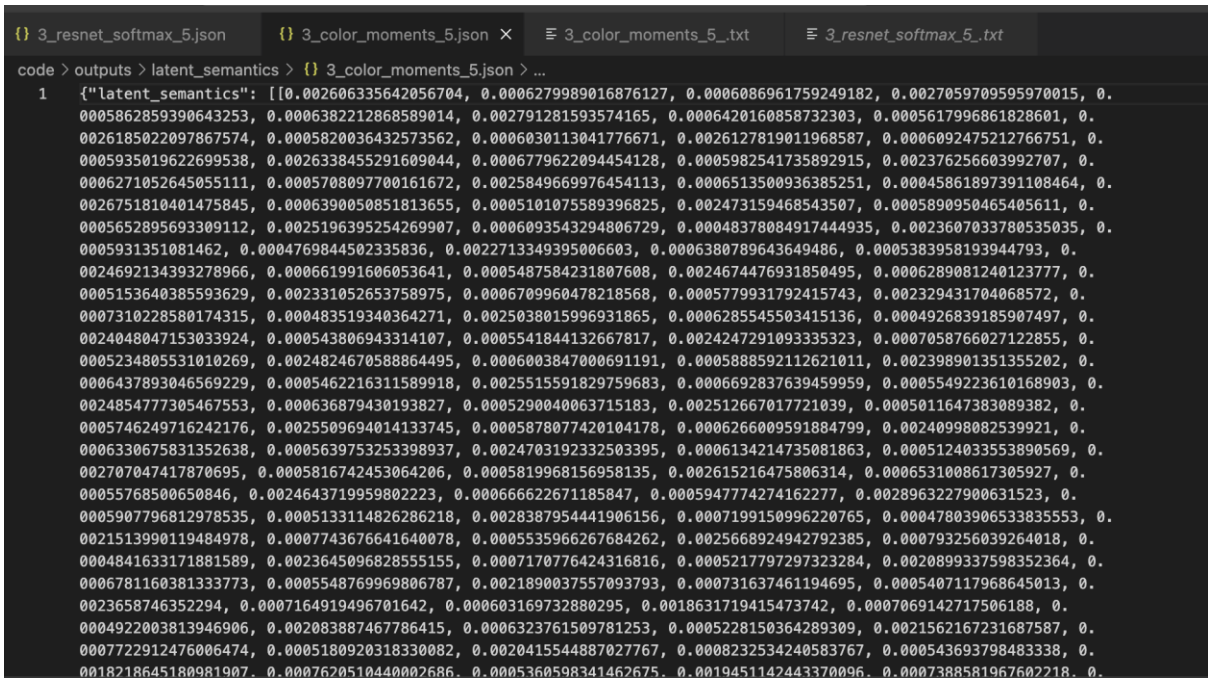

Fig 3.2: Latent Semantics for Color Moments, k=5

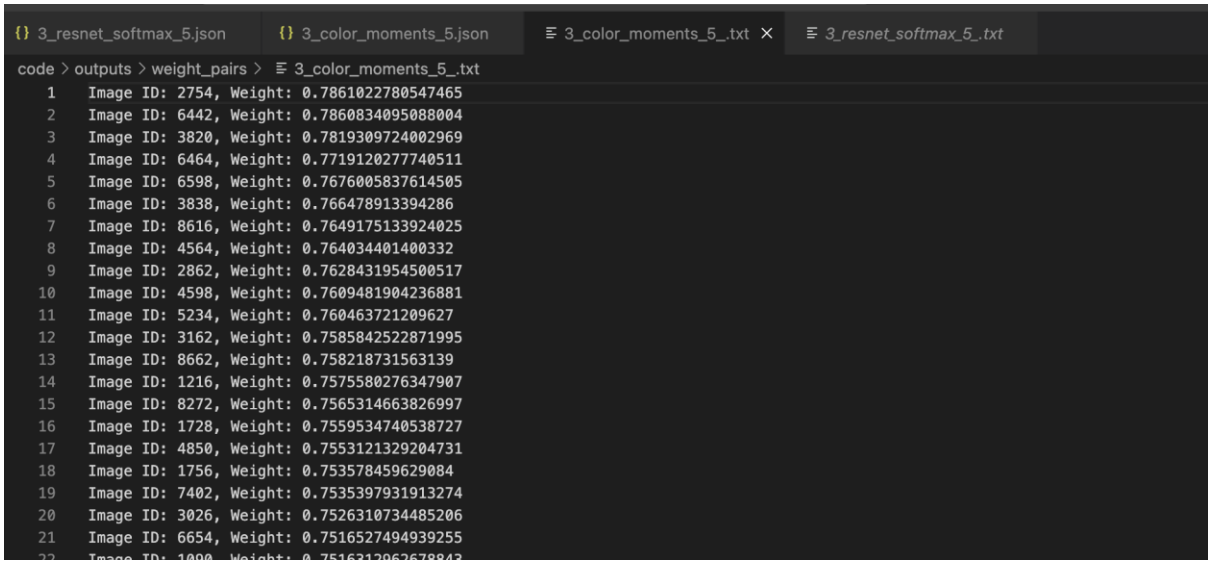

Fig 3.3: Image ID Weight Pairs for Color Moments, k=5

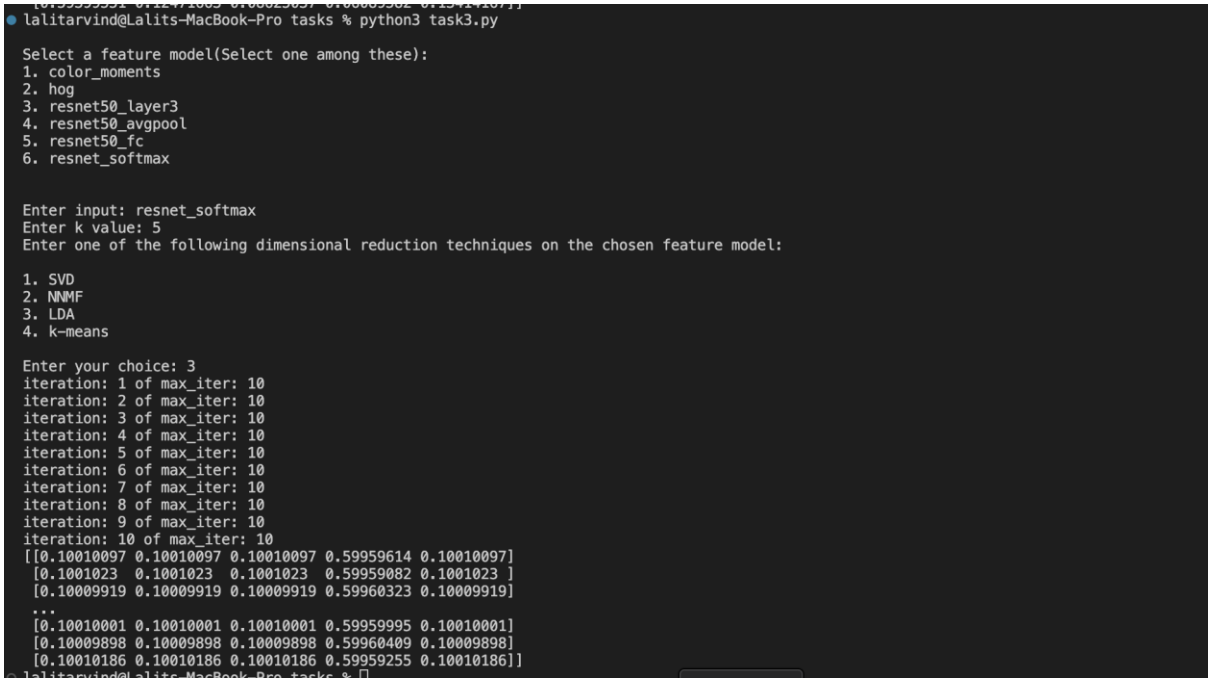

Fig 3.4: LDA on Resnet, k=5

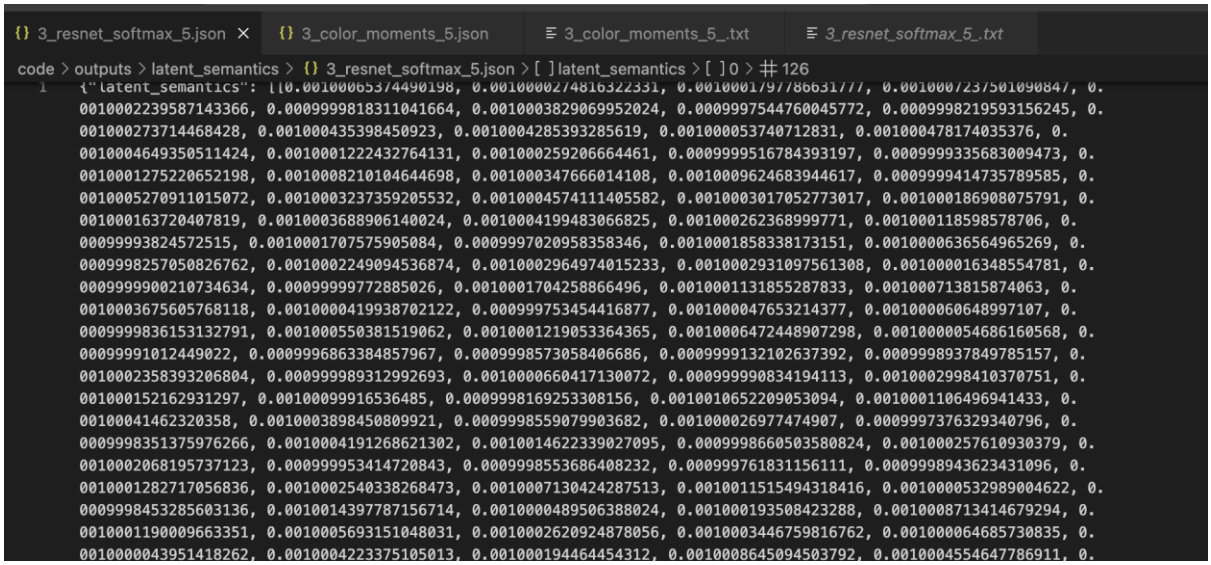

Fig 3.5: Latent Semantics for Resnet, k=5

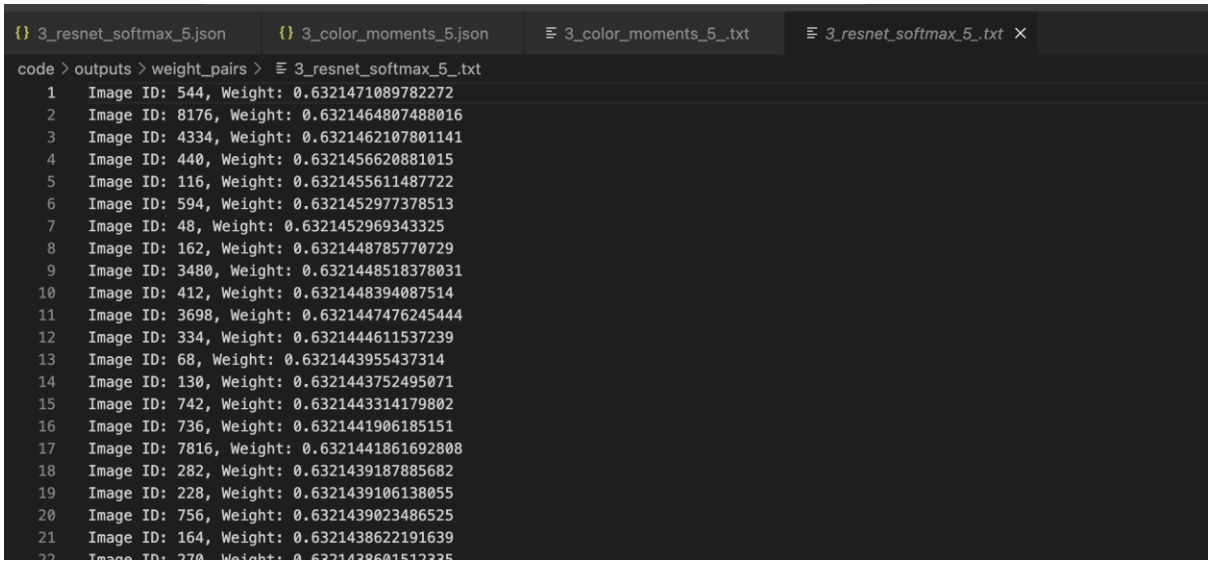

Fig 3.6: Image ID Weight Pairs for Resnet, k=5

d. Assumptions & Observations & Improvements: L2 norm of reduced features of the image are assumed to be the weight of the image.

K-MEANS:

a. Design Choices & Decisions:
   1. Centroid Initialisation: The methodology utilised for centroid initialization is the k-means centroid initialization wherein, the initial vector representing the centroids is chosen randomly and the subsequent vectors based on the probability, calculated based on the distance of the vector. The probability of choosing the subsequent vectors is inversely proportional to the distance of the point from the initial randomly chosen centroid. The motive behind selecting this method for the purpose of centroid initialization is to ensure that the initial centroids chosen mitigates the risk of k-means clustering reaching a suboptimal local minimum of the within cluster sum of squares. Utilizing this method improves the efficiency of the clustering, thereby, reducing the clustering runtime as well as reducing the number of iterations the algorithm has to go through before reaching a global minimum of within cluster sum of squares.
   2. Distance between vectors: In order to calculate the distance between vectors, Euclidean distance measure is used due to its suitability within vector spaces in the data. Moreover, it also possesses fundamental properties of a reliable distance measure, such as non-negativity, zero identity, symmetry and triangular inequality.
   3. Stopping Condition: The stopping condition in this approach is based on the convergence of centroids, specifically when the newly computed centroids from the clusters are almost the same as that of the previous iteration of clusters. Here, a tolerance condition is defined with a threshold of 10-5 ensuring that the algorithm terminates when subsequent iterations yield negligible difference in the computed centroids.

b. Expected Outcomes: Expected final centroids representing the cluster centres. Expected cluster assignments for each data vector. Convergence state is to be reached given the algorithm iterates until the difference between centroids is negligible.

c. Outputs:

```
dasar@Khadyu MINGW64 /c/Khadyu/ASU/Fall 2023/Multimedia & Web Databases/Project/phase2_project/code/tasks
$ python task3.py

Select a feature model(Select one among these):
1. color_moments
2. hog
3. resnet50_layer3
4. resnet50_avgpool
5. resnet50_fc
6. resnet50_softmax

Enter input: color_moments
Enter k value: 5
Enter one of the following dimensional reduction techniques on the chosen feature model:

1. SVD
2. NNMF
3. LDA
4. k-means

Enter your choice: 4
shape of centroids: (5, 900)
```

Fig 3.7: KMEANS on Color Moments, k=5

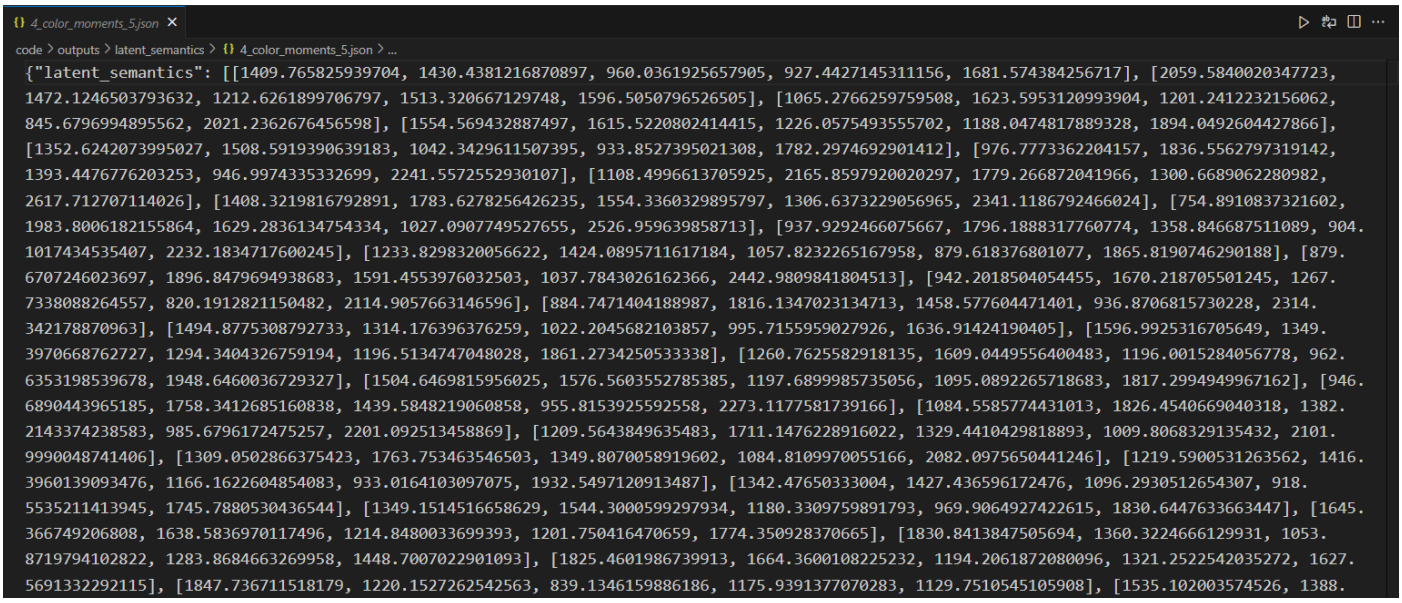

Fig 3.8: Latent Semantics for Color Moments, k=5

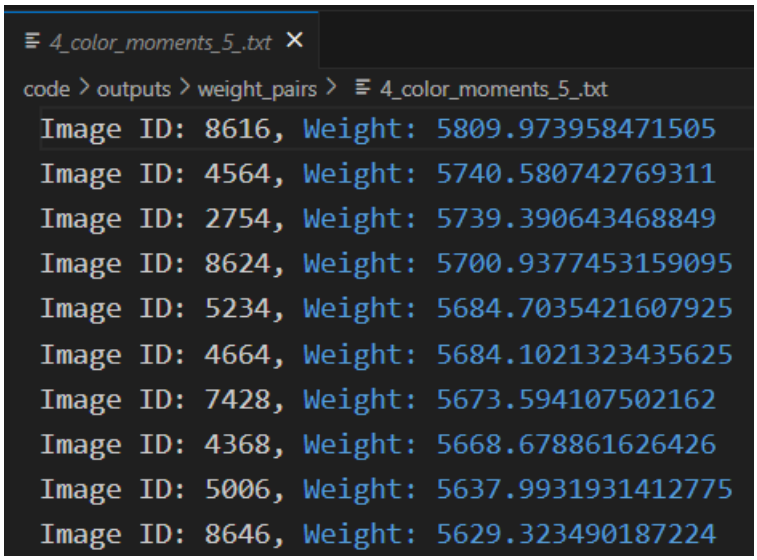

Fig 3.9: Image ID weight pairs for CM, k=5

```
dasar@Khadyu MINGW64 /c/Khadyu/ASU/Fall 2023/Multimedia & Web Databases/Project/phase2_project/code/tasks
$ python task3.py

Select a feature model(Select one among these):
1. color_moments
2. hog
3. resnet50_layer3
4. resnet50_avgpool
5. resnet50_fc
6. resnet50_softmax

Enter input: resnet50_fc
Enter k value: 10
Enter one of the following dimensional reduction techniques on the chosen feature model:

1. SVD
2. NNMF
3. LDA
4. k-means

Enter your choice: 4
shape of centroids: (10, 1000)
```

Fig 3.10: KMEANS on Resnet FC, k=5

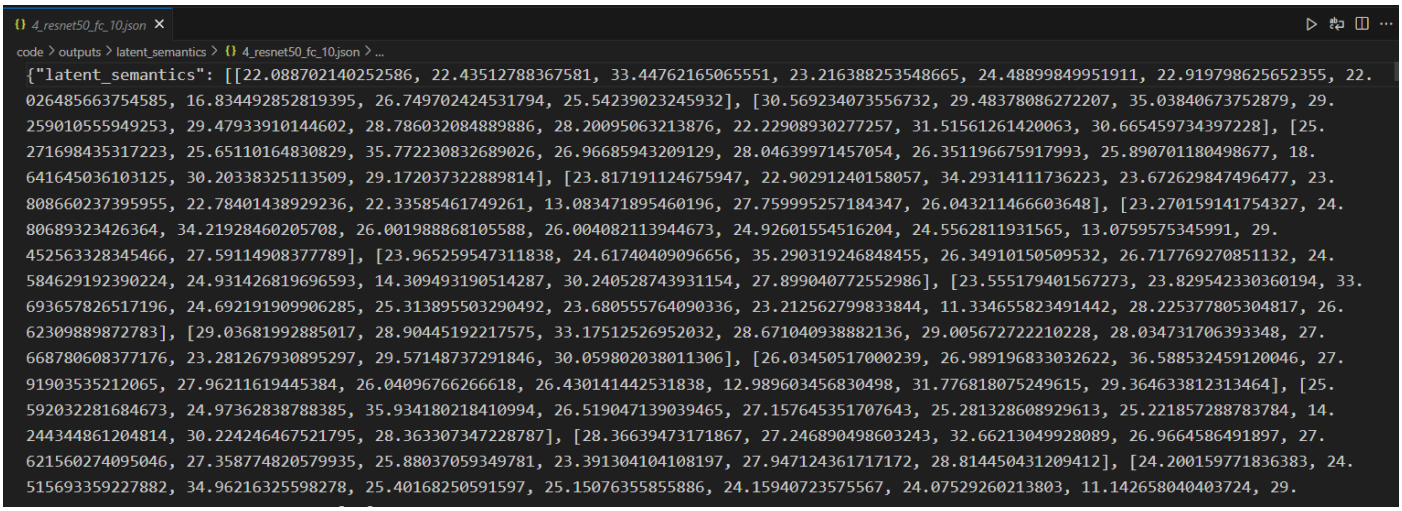

Fig 3.11: Latent Semantics for Resnet FC, k=5

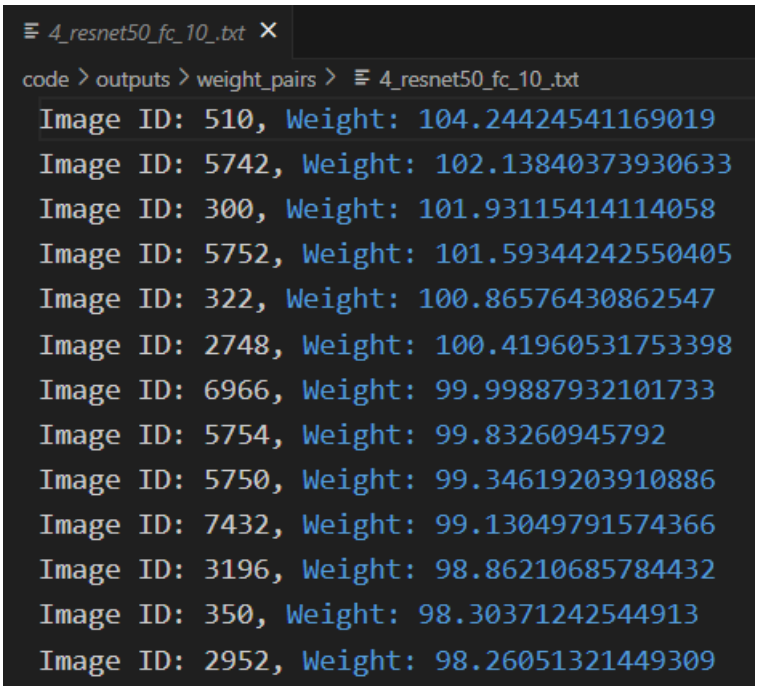

Fig 3.12: Image ID weight pairs for Resnet FC, k=5

d. Assumptions & Observations & Improvements: Convergence is achieved based on the fact whether the difference between centroids of iterations is negligible. Initialization of centroids is using k-means initialization. However, we can find an alternative method for centroid initialization to enhance cluster performance and accuracy and can implement batch processing of the data matrix for faster execution.

Task 4: Dimensionality Reduction (CP Decomposition)

Query Input Given:

User gives the feature model and k. Need to store the top-k latent semantics after performing the CP Decomposition on the given feature model and for each latent semantic list the label weight pairs in descending order of weights.

Implementation:

The data matrix (4339 * 900 * 101) or its reduced form (4339 * 900 * 1) is created based on the feature model that the user selects. The program uses CP decomposition with the specified rank k to extract latent semantics. It calculates label-weight pairs for each latent semantic by taking the average of all images for each label in the latent space.

Design Choices & Decisions:

PARAFAC library is used for performing CP decomposition. PARAFAC is used for its efficiency, parallelisation, scalability and many other advantages it provides with. To handle the tensor size being too large and time-consuming error, we reduced the last mode to 1 and performed the decomposition (4339x900x1)

Expected Outcomes:

When CP decomposition is performed on a tensor of shape 4339 * 900 * 101, essentially the latent semantic information is extracted from a multi-modal dataset.

Latent Semantics (Factors): The primary outcome of CP decomposition is a set of factors matrices. These factors represent patterns and relationships in the data. These factors can be interpreted as follows:

- The first factor represents images in terms of new features.
- The second factor represents features in terms of new features.
- The third factor represents labels in terms of new features.

Label-Weight Pairs: Using the first factor matrix (representing images in terms of new features), you can calculate label-weight pairs for each latent semantic. These pairs represent the relationship between labels and the discovered latent features.

Outputs:

```
dasar@Khadyu MINGW64 /c/Khadyu/ASU/Fall 2023/Multimedia & Web Databases/Project/phase2_project/code/tasks
$ python task4.py

Select a feature model(Select one among these):
1. color_moments
2. hog
3. resnet50_layer3
4. resnet50_avgpool
5. resnet50_fc
6. resnet50_softmax

Enter input: hog
Enter k value: 5
(4339, 900)
```

Fig 4.1: CP on HOG, k=5

```
ls_dict = {
    "image_image": latent_semantics[0].tolist(),
    "feature_feature": latent_semantics[1].tolist(),
    "label_label": latent_semantics[2].tolist(),
}
```

Fig 4.2: Latent Semantics Json Structure

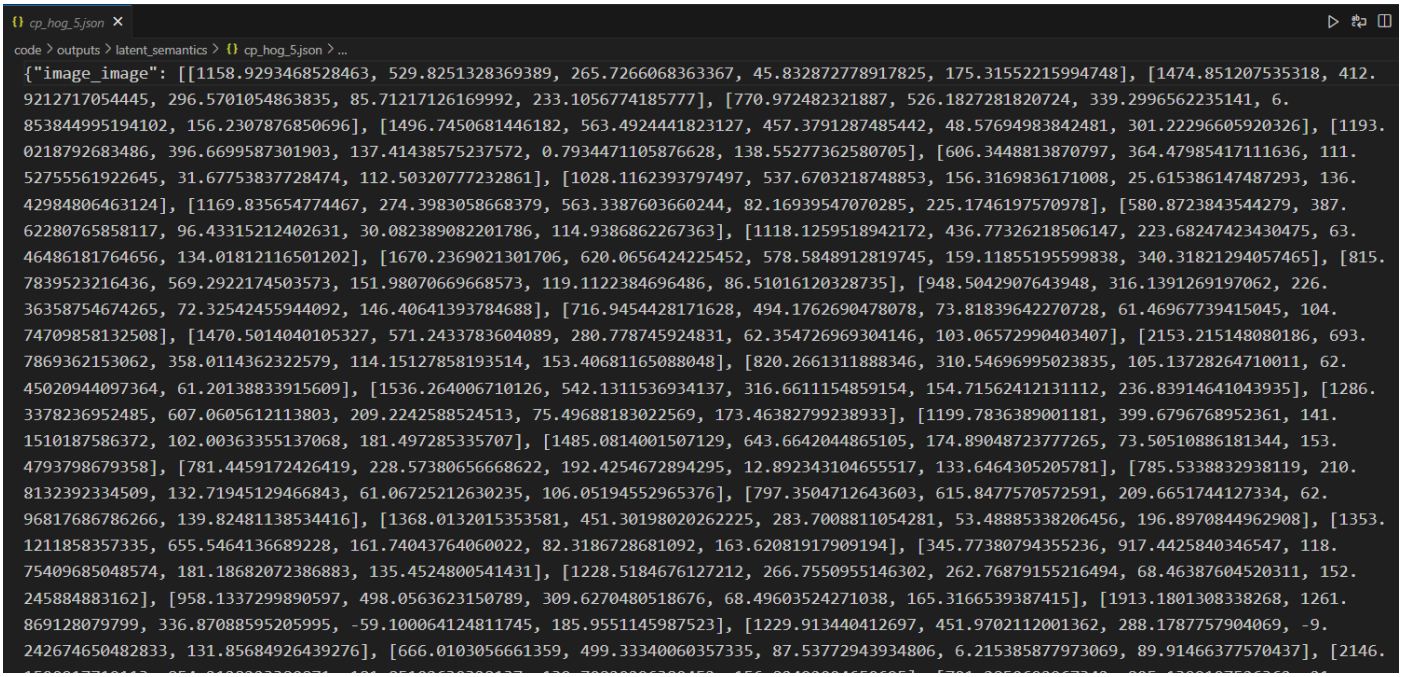

Fig 4.3: Latent Semantics of HOG, k=5

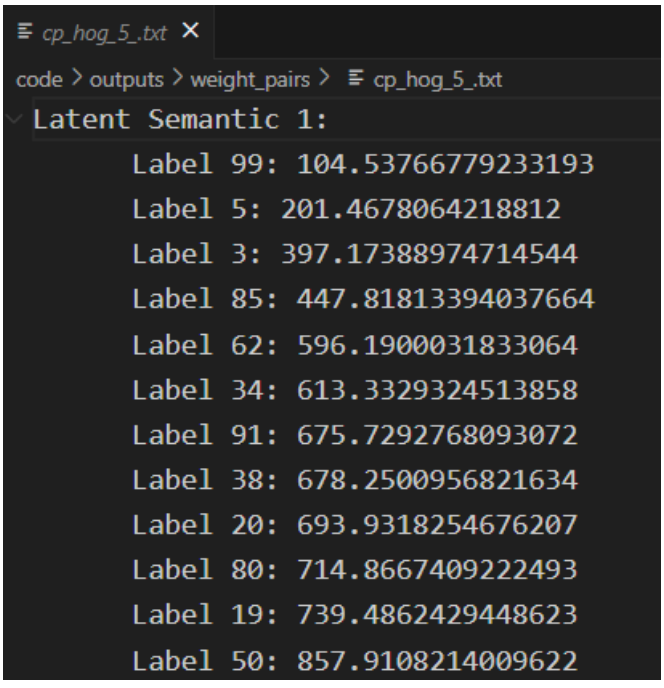

Fig 4.4: Label weight pairs for HOG, k=5

```
dasar@Khadyu MINGW64 /c/Khadyu/ASU/Fall 2023/Multimedia & Web Databases/Project/phase2_project/code/tasks
$ python task4.py

Select a feature model(Select one among these):
1. color_moments
2. hog
3. resnet50_layer3
4. resnet50_avgpool
5. resnet50_fc
6. resnet50_softmax

Enter input: resnet50_softmax
Enter k value: 5
(4339, 1000)
```

Fig 4.5: CP on Resnet, k=5

cp_resnet50_softmax_5.json

code > outputs > latent_semantics > cp_resnet50_softmax_5.json > ...


```
{"image_image": [[-7.1882433851905e-08, -5.610806298583533e-07, 6.000036787052234e-06, -2.5619510195175085e-08, -8.433854498830782e-07], [-7.
1882433851905e-08, -5.610806298583533e-07, 6.000036787052234e-06, -2.5619510195175085e-08, -8.433854498830782e-07], [-7.1882433851905e-08, -5.
610806298583533e-07, 6.000036787052234e-06, -2.5619510195175085e-08, -8.433854498830782e-07], [-7.1882433851905e-08, -5.610806298583533e-07, 6.
000036787052234e-06, -2.5619510195175085e-08, -8.433854498830782e-07], [-7.1882433851905e-08, -5.610806298583533e-07, 6.000036787052234e-06, -2.
5619510195175085e-08, -8.433854498830782e-07], [-7.1882433851905e-08, -5.610806298583533e-07, 6.000036787052234e-06, -2.5619510195175085e-08,
-8.433854498830782e-07], [-7.1882433851905e-08, -5.610806298583533e-07, 6.000036787052234e-06, -2.5619510195175085e-08, -8.
433854498830782e-07], [-7.1882433851905e-08, -5.610806298583533e-07, 6.000036787052234e-06, -2.5619510195175085e-08, -8.433854498830782e-07],
[-7.1882433851905e-08, -5.610806298583533e-07, 6.000036787052234e-06, -2.5619510195175085e-08, -8.433854498830782e-07], [-7.1882433851905e-08,
-5.610806298583533e-07, 6.000036787052234e-06, -2.5619510195175085e-08, -8.433854498830782e-07], [-7.1882433851905e-08, -5.610806298583533e-07,
6.000036787052234e-06, -2.5619510195175085e-08, -8.433854498830782e-07], [-7.1882433851905e-08, -5.610806298583533e-07, 6.000036787052234e-06,
-2.5619510195175085e-08, -8.433854498830782e-07], [-7.1882433851905e-08, -5.610806298583533e-07, 6.000036787052234e-06, -2.
5619510195175085e-08, -8.433854498830782e-07], [-7.1882433851905e-08, -5.610806298583533e-07, 6.000036787052234e-06, -2.5619510195175085e-08,
-8.433854498830782e-07], [-7.1882433851905e-08, -5.610806298583533e-07, 6.000036787052234e-06, -2.5619510195175085e-08, -8.
433854498830782e-07], [-7.1882433851905e-08, -5.610806298583533e-07, 6.000036787052234e-06, -2.5619510195175085e-08, -8.433854498830782e-07],
[-7.1882433851905e-08, -5.610806298583533e-07, 6.000036787052234e-06, -2.5619510195175085e-08, -8.433854498830782e-07], [-7.1882433851905e-08,
-5.610806298583533e-07, 6.000036787052234e-06, -2.5619510195175085e-08, -8.433854498830782e-07], [-7.1882433851905e-08, -5.610806298583533e-07,
6.000036787052234e-06, -2.5619510195175085e-08, -8.433854498830782e-07], [-7.1882433851905e-08, -5.610806298583533e-07, 6.000036787052234e-06,
-2.5619510195175085e-08, -8.433854498830782e-07], [-7.1882433851905e-08, -5.610806298583533e-07, 6.000036787052234e-06, -2.
5619510195175085e-08, -8.433854498830782e-07], [-7.1882433851905e-08, -5.610806298583533e-07, 6.000036787052234e-06, -2.5619510195175085e-08,
-8.433854498830782e-07], [-7.1882433851905e-08, -5.610806298583533e-07, 6.000036787052234e-06, -2.5619510195175085e-08, -8.
433854498830782e-07], [-7.1882433851905e-08, -5.610806298583533e-07, 6.000036787052234e-06, -2.5619510195175085e-08, -8.433854498830782e-07]
```


Fig 4.6: Latent Semantics of Resnet, k=5

cp_resnet50_softmax_5_.txt

code > outputs > weight_pairs > cp_resnet50_softmax_5_.txt

```
Latent Semantic 1:
        Label 100: -2.084590581705246e-06
        Label 2: -7.188243385190512e-08
        Label 94: -7.188243385190507e-08
        Label 4: -7.188243385190504e-08
        Label 8: -7.188243385190504e-08
        Label 9: -7.188243385190504e-08
        Label 10: -7.188243385190504e-08
        Label 17: -7.188243385190504e-08
        Label 20: -7.188243385190504e-08
        Label 24: -7.188243385190504e-08
        Label 28: -7.188243385190504e-08
        Label 29: -7.188243385190504e-08
        Label 30: -7.188243385190504e-08
        Label 32: -7.188243385190504e-08
```

Fig 4.7: Label weight pairs for Resnet, k=5

Assumptions & Observations & Improvements:

We assume that the 4339 * 900 (for Color_moments, HOG) and (4339 * 1024 or 4339 * 1000 for ResNet) data matrix is already stored in the project folder.

Task 5: Dimensionality Reduction on Label-Label Similarity Matrix

Query Input Given:

Given a feature space as an input, we need to perform task3 on the label-label similarity matrix.

Implementation:

The given input feature models label-label-similarity-matrix is created. This will be a (101x101) matrix since there are a total of 101 labels. This matrix is then used as the input to task3 the rest of the implementation is same as task3. Additionally, the naming convention of the latent semantics and label-weight pairs are same as on task3 but appended with a `_llsm` in the end indicating it belongs to label-label-similarity matrix.

Design Choices & Decisions:

All the label-label-similarity matrices are generated using a python script. Cosine similarity is used to generate the matrix. Cosine similarity is used since the labels are sensitive to data distribution and using distance metrics might not give accurate results.

Expected Outcomes:

The latent semantics of the given label-label similarity matrix is generated by taking the input feature model, k and dimensionality reduction technique from the user.

Outputs:

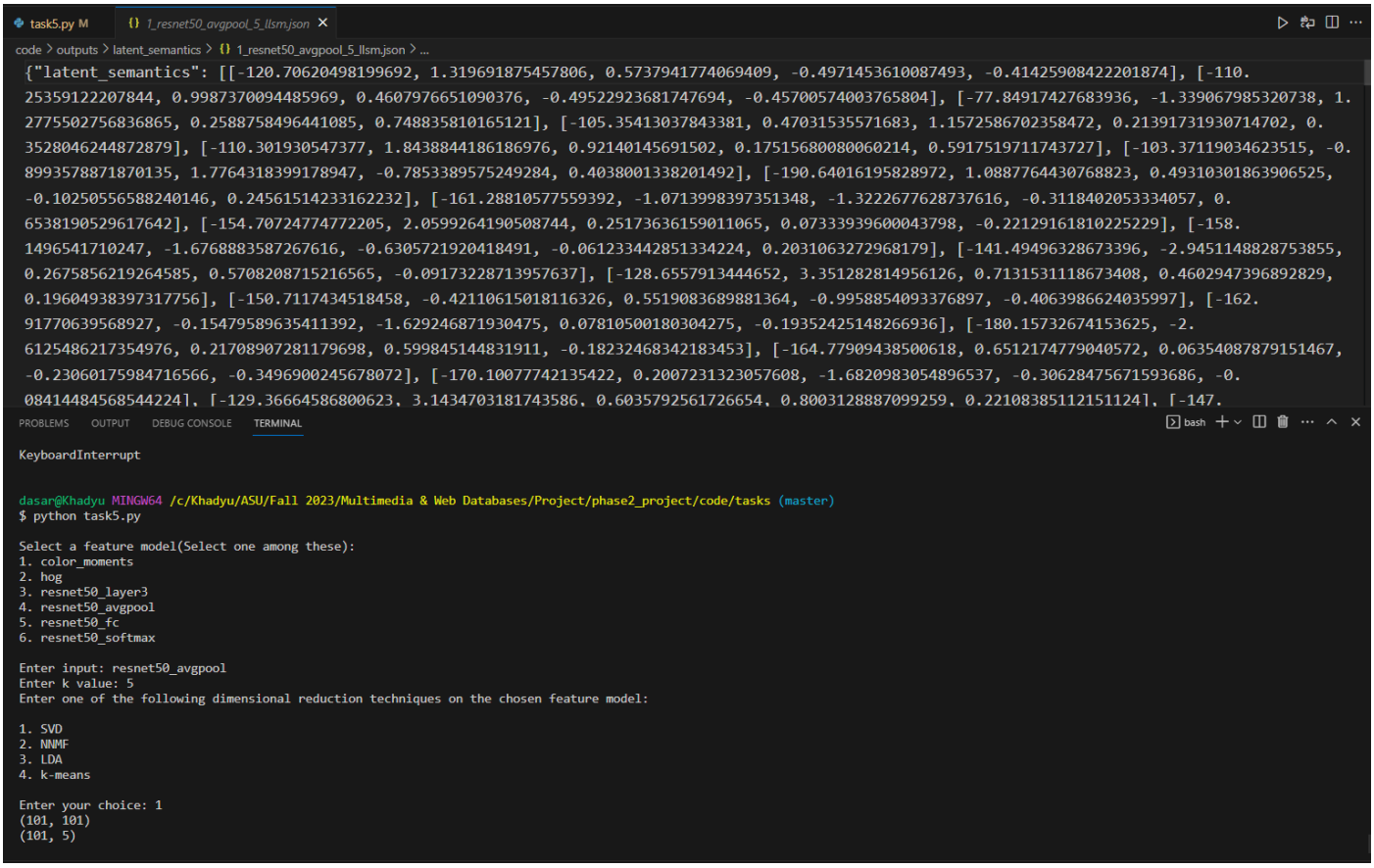

Fig 5.1: AvgPool, 5, SVD

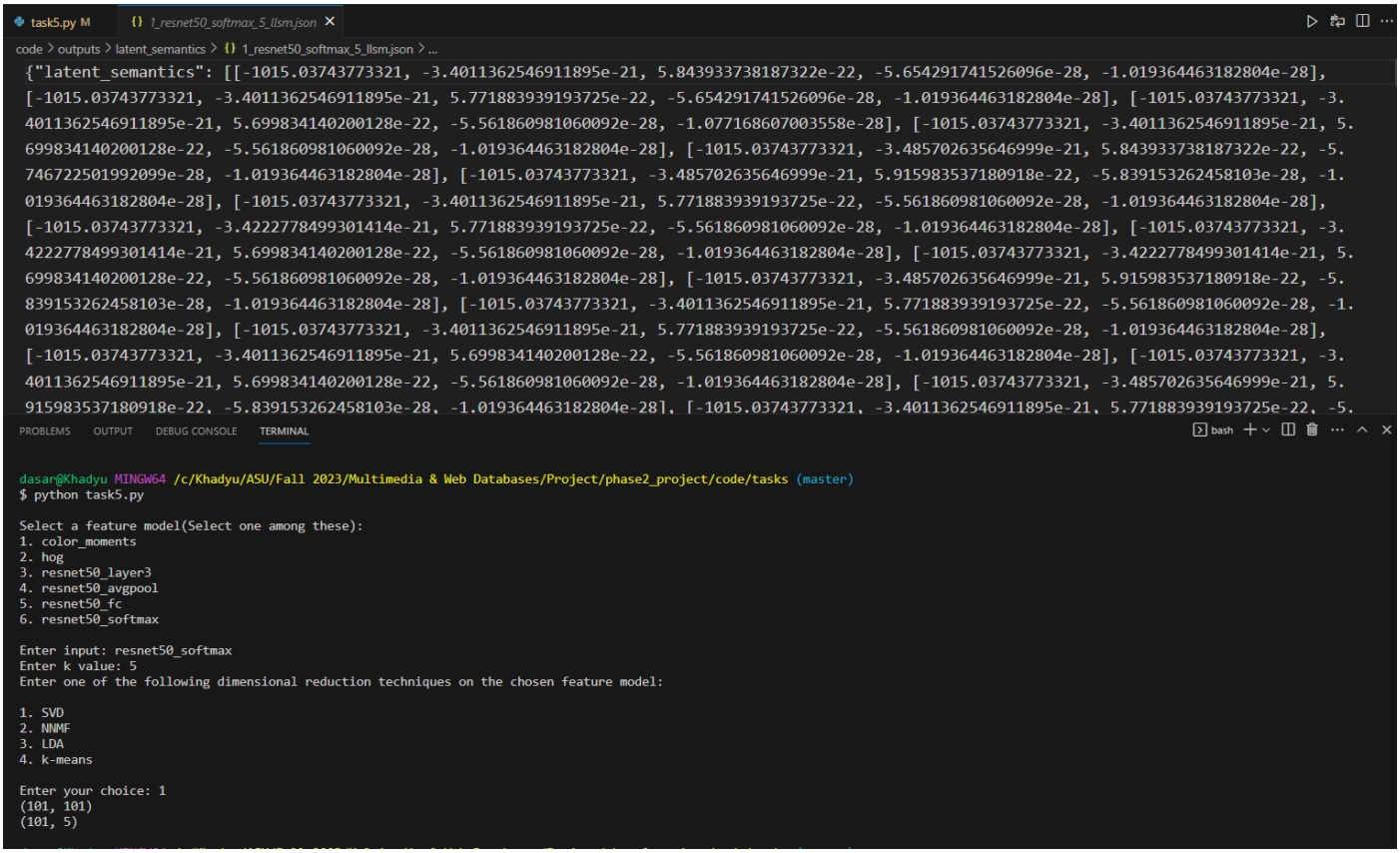

Fig 5.2: Resnet, 5, SVD

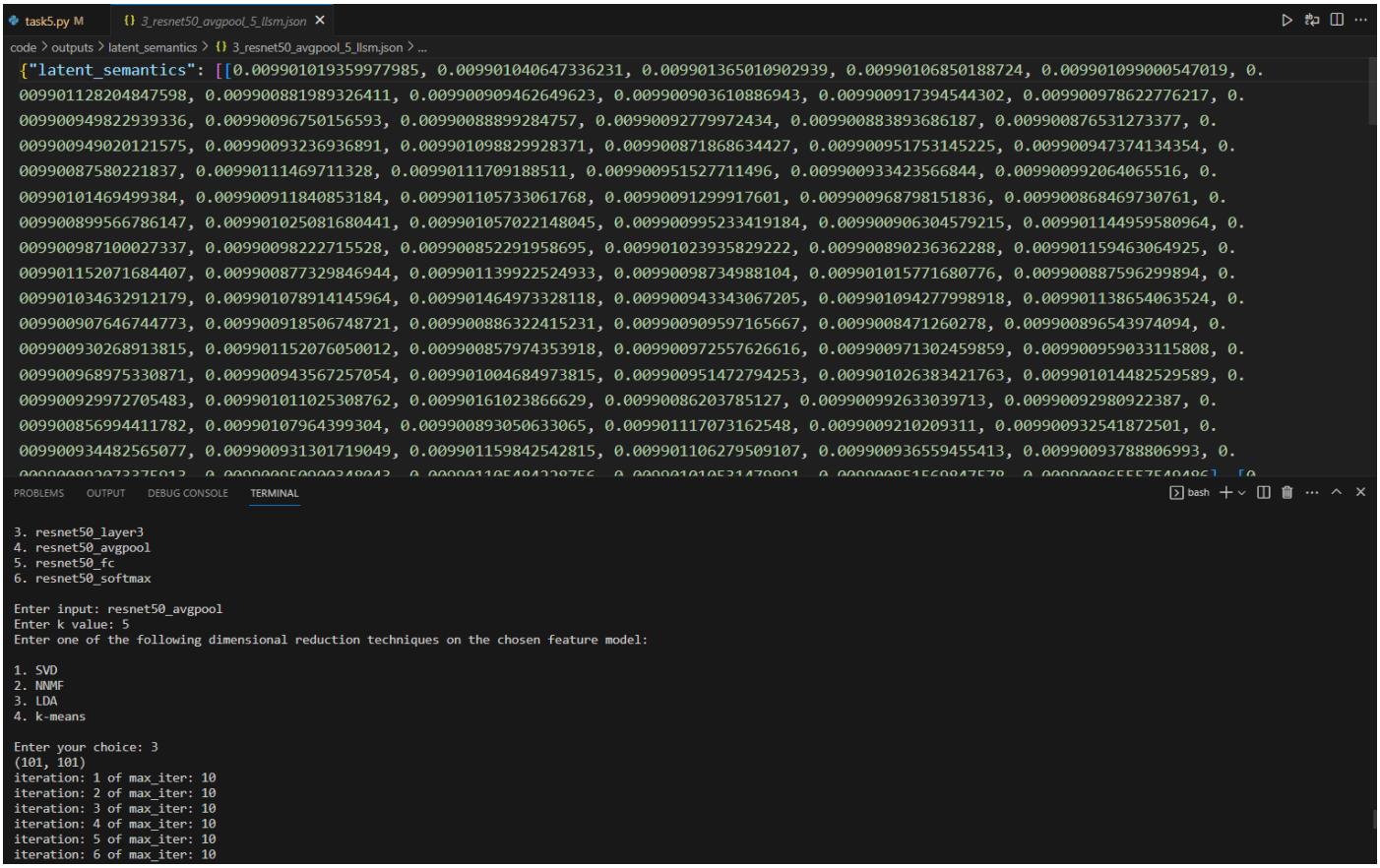

Fig 5.3: AvgPool, 5, LDA

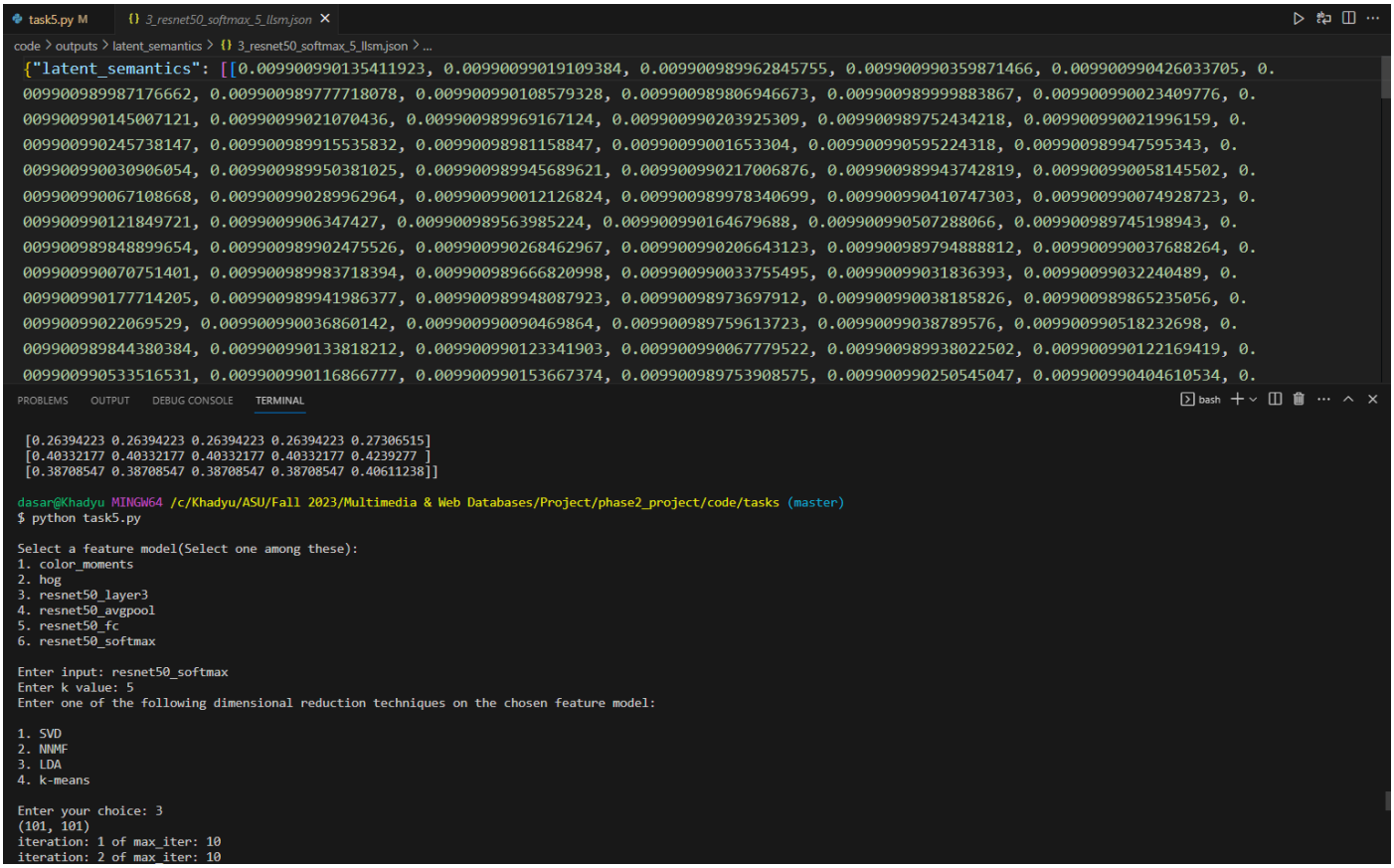

Fig 5.4: Resnet, 5, LDA

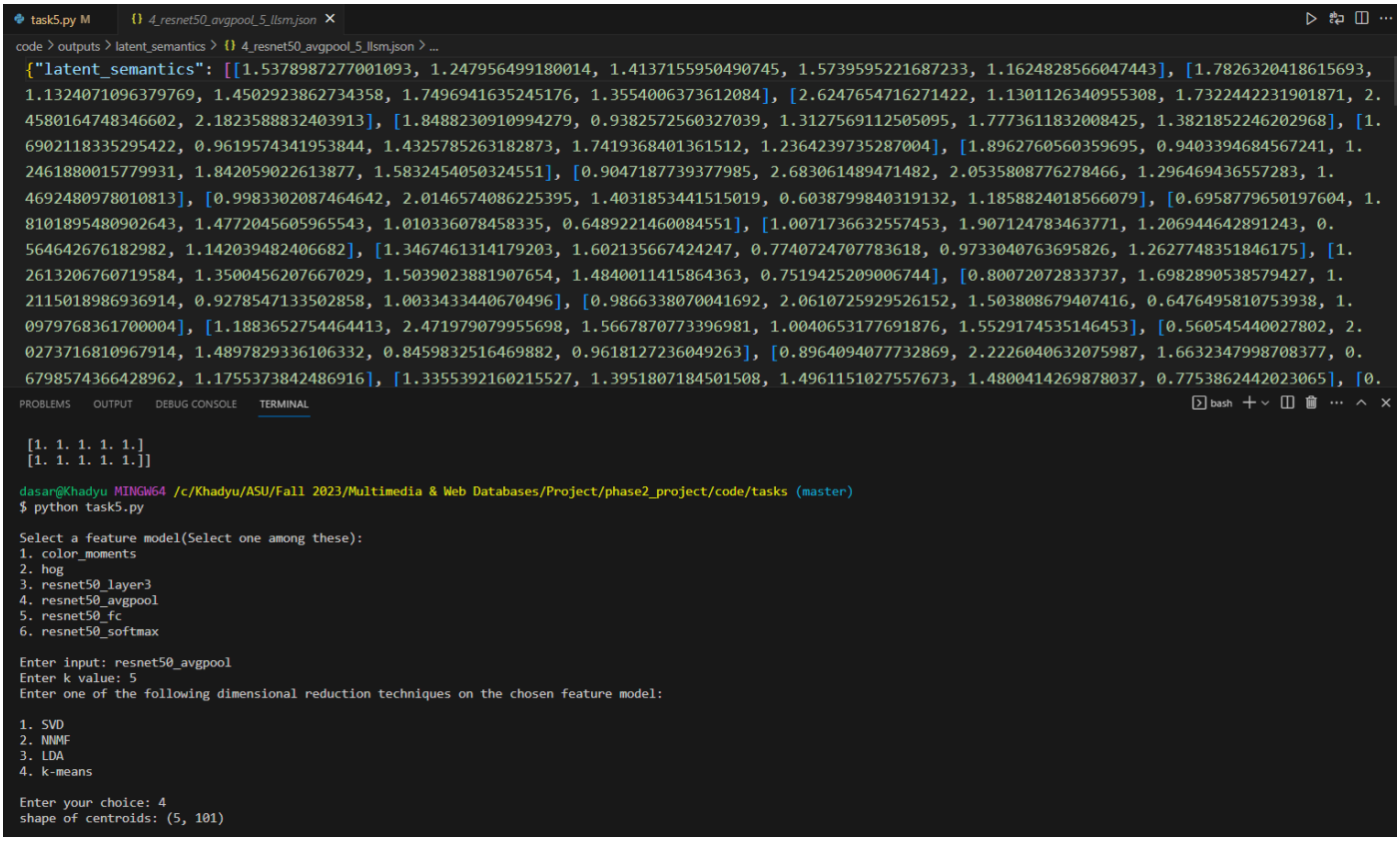

Fig 5.5: AvgPool, 5, KMEANS

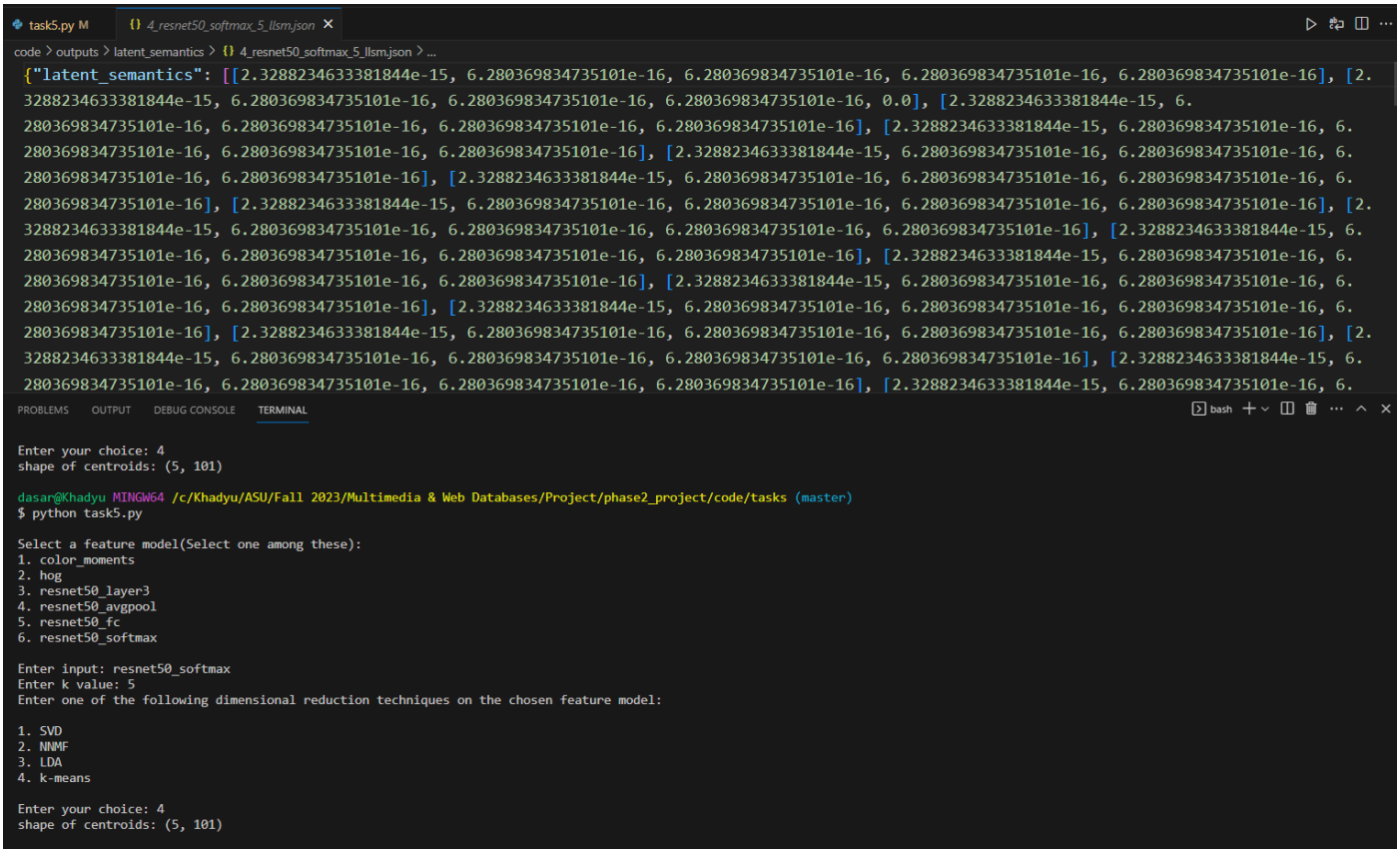

Fig 5.6: Resnet, 5, KMEANS

Assumptions & Observations & Improvements:

Cosine similarity is used to generate the matrix. Cosine similarity is used since the labels are sensitive to data distribution and using distance metrics might not give accurate results. NNMF gives similar results to SVD since both are decompositions, but with a added constraint.

Task 6: Dimensionality Reduction on Image-Image Similarity Matrix

Query Input Given:

Given a feature space as an input, we need to perform task3 on the image-image similarity matrix.

Implementation:

The given input feature models image-image similarity matrix is created. This will be a (4339x4339) matrix since there are a total of 4339 images. This matrix is then used as the input to task3 the rest of the implementation is same as task3. Additionally, the naming convention of the latent semantics and label-weight pairs are same as on task3 but appended with a `_iism` in the end indicating it belongs to image-image similarity matrix.

Design Choices & Decisions:

Since computing the image-image similarity matrix requires huge amount of time, all the image-image similarity matrices are generated using a python script. Cosine similarity is used to generate the matrix. Cosine similarity is used since the images might be sensitive to data distribution and using distance metrics might not give accurate results.

Expected Outcomes:

The latent semantics of the given image-image similarity matrix is generated by taking the input feature model, k and dimensionality reduction technique from the user.

Outputs:

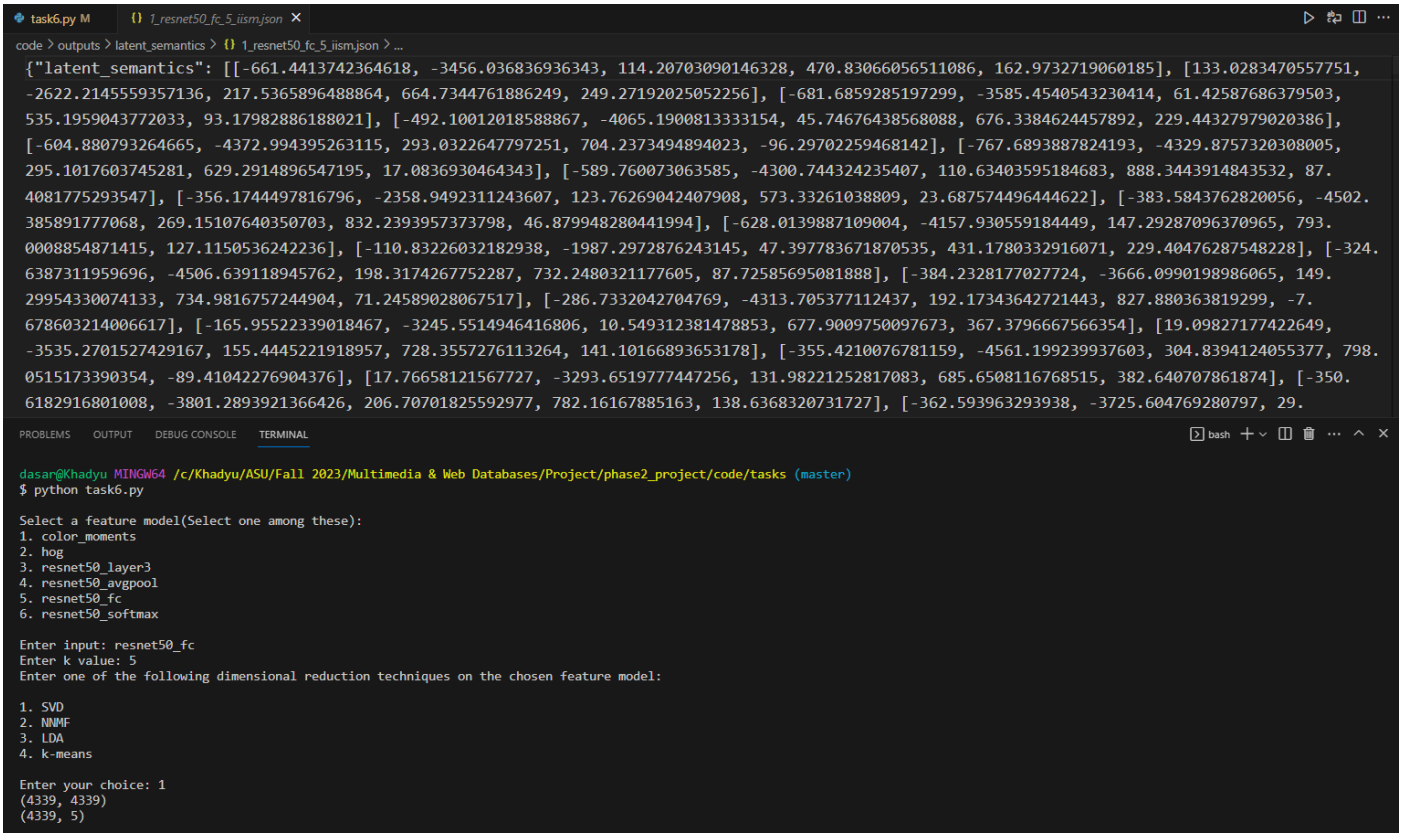

Fig 6.1: FC, 5, SVD

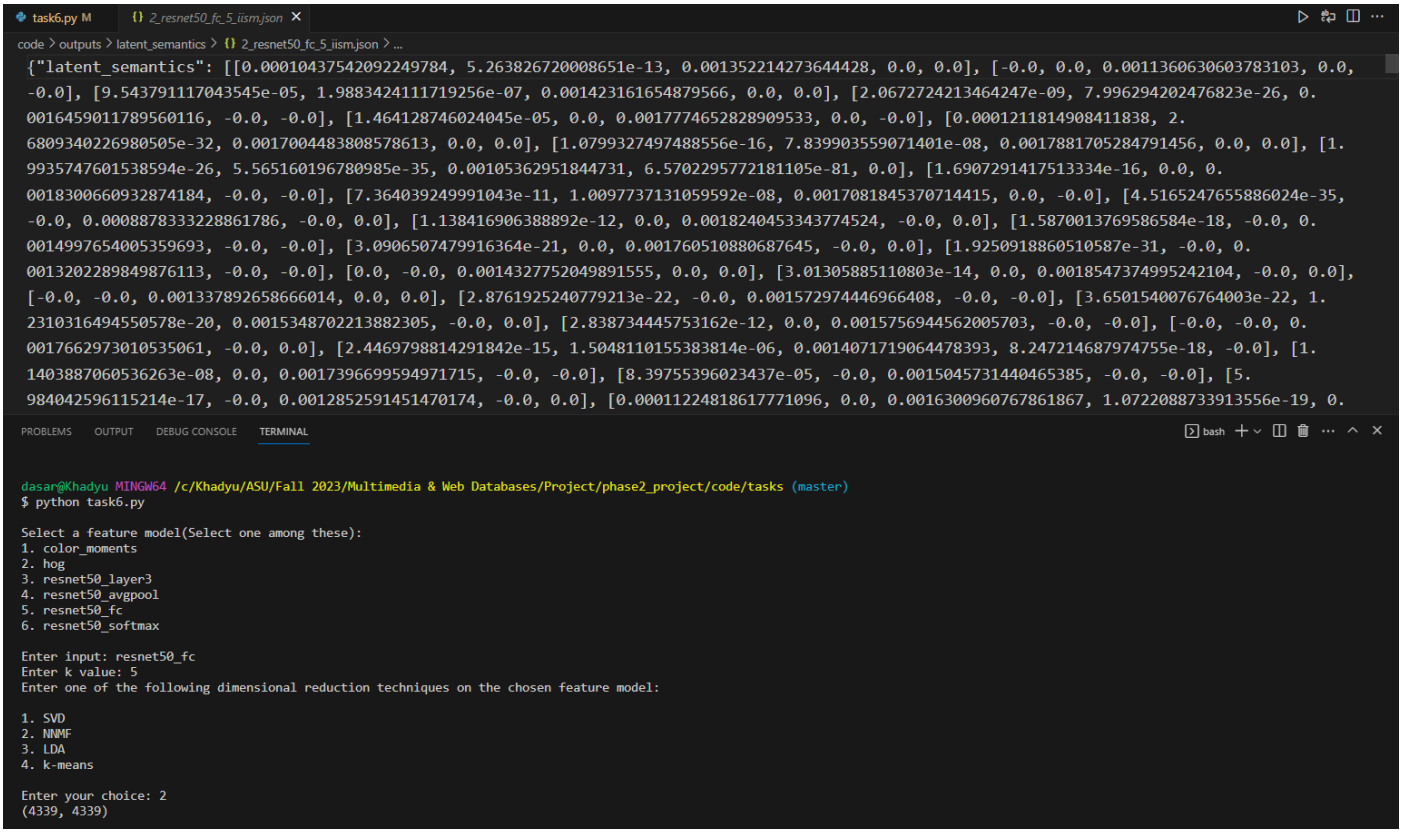

Fig 6.2: FC, 5, NNMF

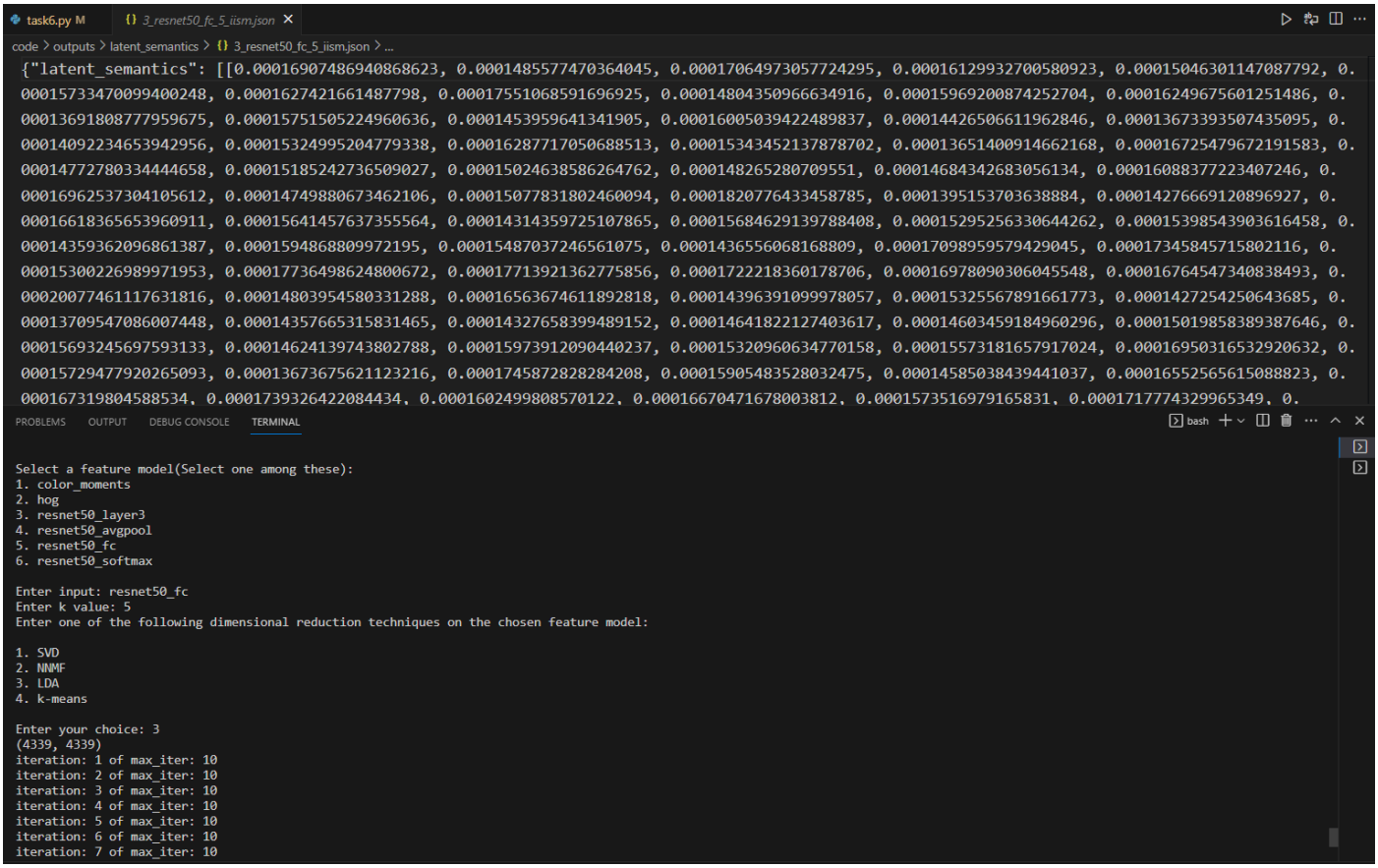

Fig 6.3: FC, 5, LDA

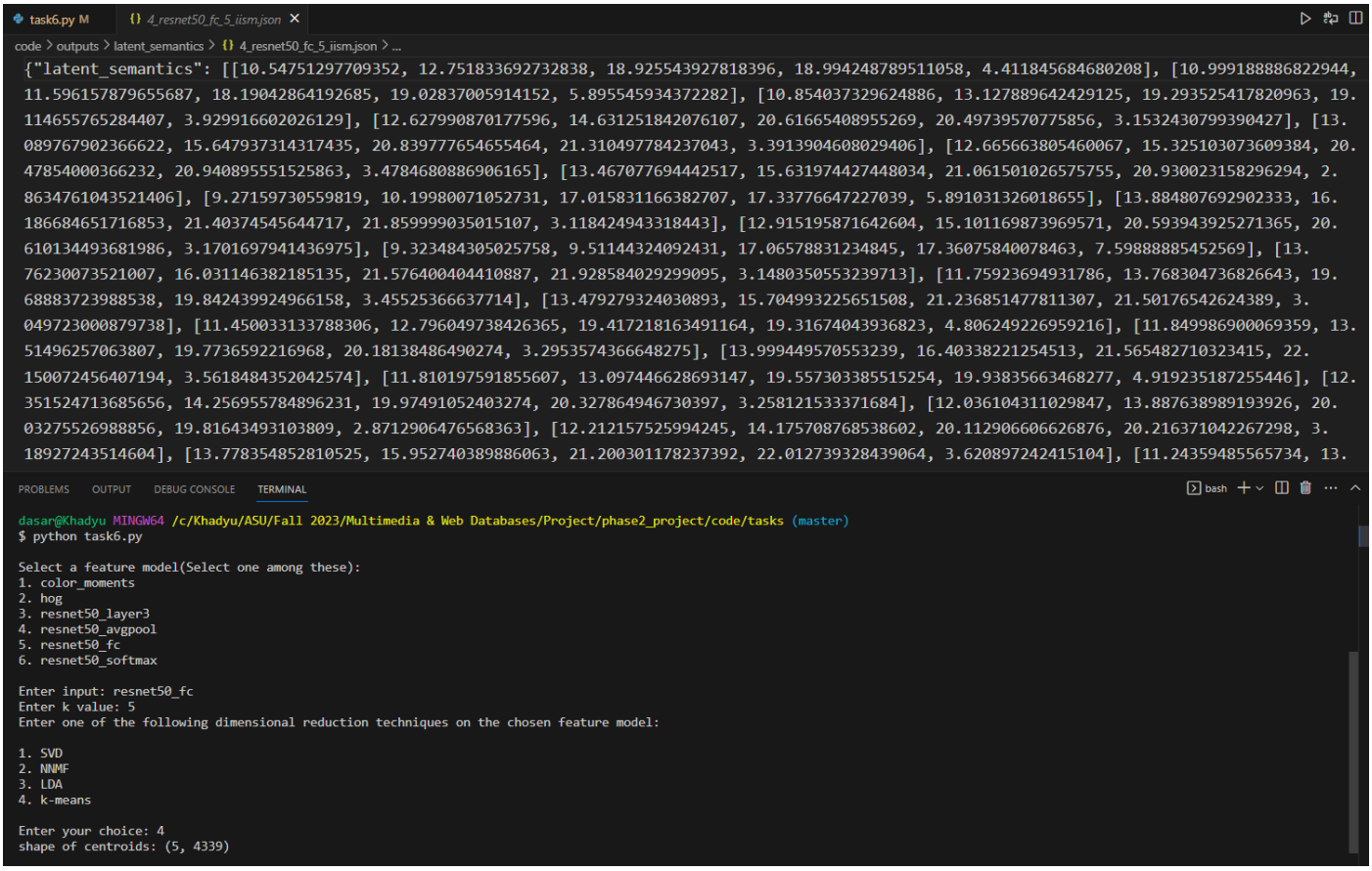

Fig 6.4: FC, 5, KMEANS

Assumptions & Observations & Improvements:

Cosine similarity is used to generate the matrix. It is also observed that LDA and KMEANS are taking much time to run the dimensionality on image-image similarity data. The 4339x4339 dimensional matrix might be

the reason for it. Highly computation expensive. Out of KMEANS and LDA, LDA takes the highest time to run on the data.

Task 7: Find K Similar Images given a Query Image using Latent Semantics from Task 3-6

Query Input Given:
Given an image id as an input, need to perform task 3 to task 6 based on the user query and visualize k similar images from the new latent space.

Implementation:
Task3 to Task6 generate the latent semantics of the feature models given as an input. The image is used to find the top k similar images in this latent semantics.

Design Choices & Decisions:
T3 & T4 &T6: Since the latent semantics in task 3 & 4 returns a data matrix of shape 4339xk, top k similar images are found by performing distance analysis for the query image and the latent space.
T5: Here the latent semantics are 101xk matrix and we can't simply follow the procedure of T3-6, instead for the 101 rows the representative images are found by using the techniques used in previous tasks for getting the representative image of a label. These 101 images are used and distance-based analysis is done to get the top k similar images.

Expected Outcomes:
Given an input image-id, it performs the necessary task based on the input and visualizes top k similar images.

Outputs:
All the query inputs take K=10 (number of similar images to be printed) and k=5 (latent space dimensions)

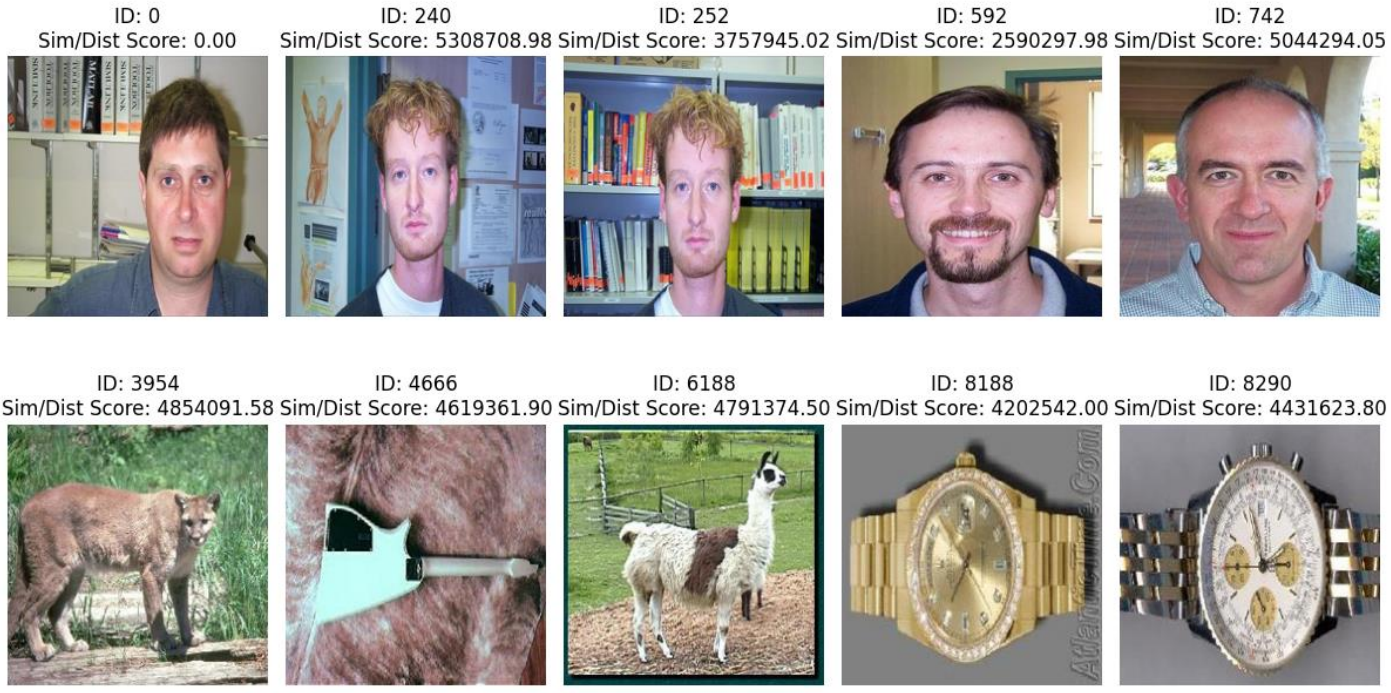

Fig 7.1: Image ID 0, T3, CM, SVD

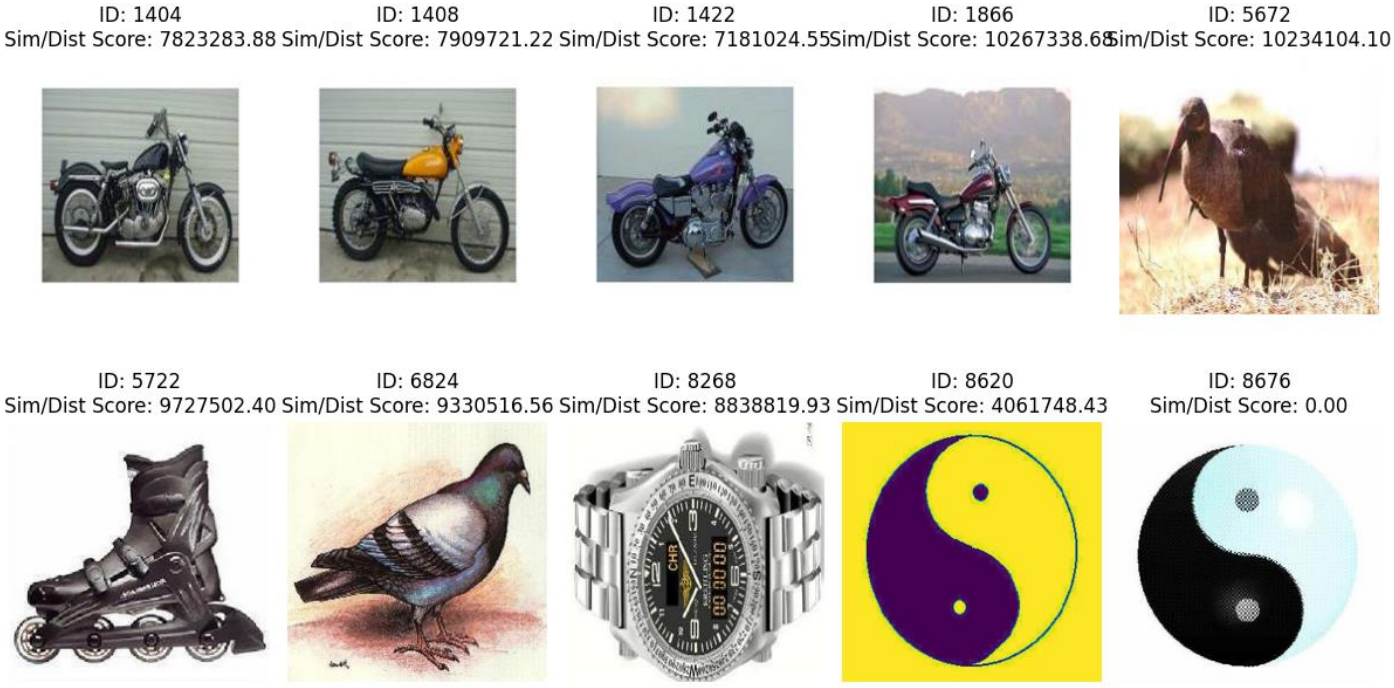

Fig 7.2: Image ID 8676, T3, CM, SVD

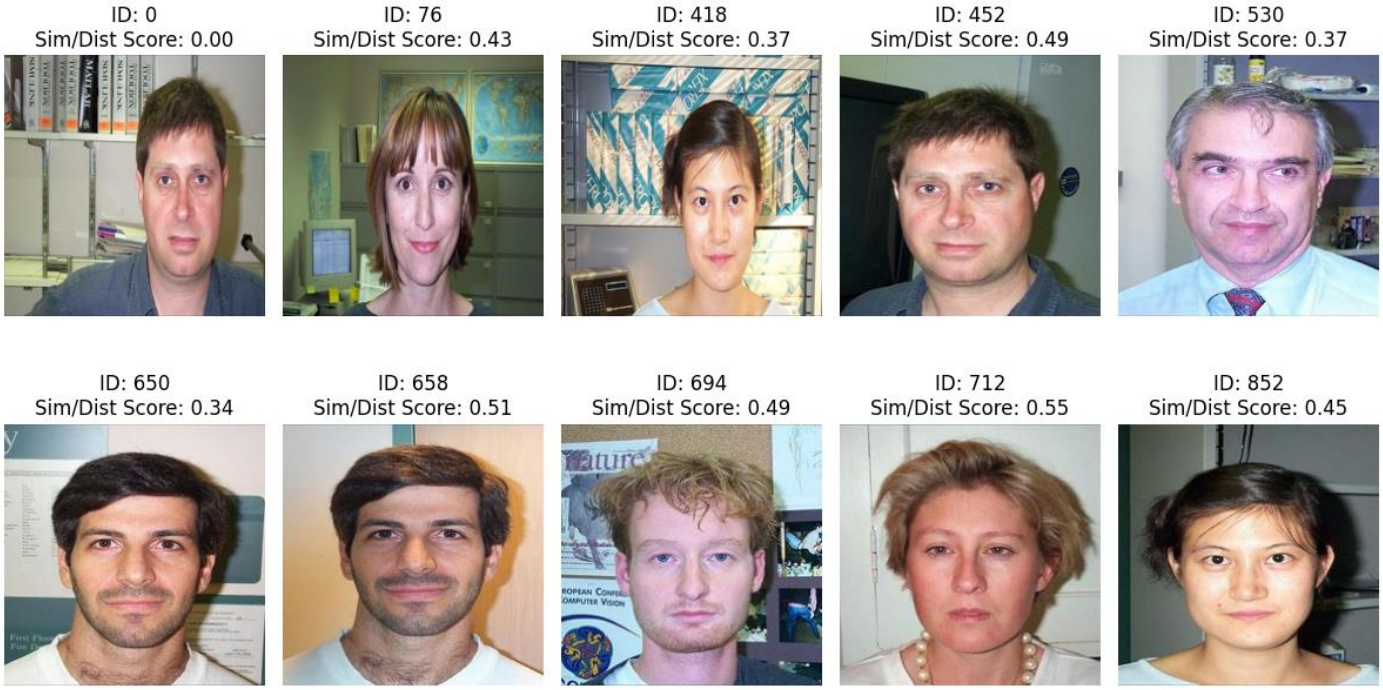

Fig 7.3: Image ID 0, T4, Resnet

Fig 7.4: Image ID 8676, T4, Resnet

Assumptions & Observations & Improvements:

We assume that the calculated representative image rightly describes the label both in latent space as well as the feature space. We assume the same for tasks 8-10 as well. From the outputs its evident that for Image ID 0, both T3 and T4 are giving better results and Resnet gives best results among the feature models. Whereas 8676, one or two similar images are showed and the distance scores in color moments are very high which might be the reason for wrong results.

Task 8: Find K Closest Labels given a Query Image using Latent Semantics from Task 3-6

Query Input Given:

User inputs the query image, required latent semantics and K closest labels.

Implementation:

The Latent Semantics are extracted as per the user's input and the category of the closest representatives in latent space are listed as the K closest labels.

Design Choices & Decisions:

1. Choosing of representative images for each label, since the representative image is the image closest to the mean of the feature it will be the most suited. To find the k closest labels which are the representative images from all the categories are selected by the following process for given feature space.
    a. Take mean of all the features of the feature vector for all the images within the same label and find the closest image to the mean image using Euclidean distance.
    b. The closest image will be utilized as the representative image for a given label and feature space.
2. Usage of JS divergence for LDA feature reduction technique because the latent semantics vector for LDA is a probabilistic distribution.

Expected Outcomes: The accurate labels of the given query image are accurately found and displayed within the K number of labels

Outputs:

```
Enter query image ID: 0
Enter any of the Latent Semantics:
1. LS1
2. LS2
3. LS3
4. LS4

Enter your choice number: 3
Enter K value for finding K similar labels: 10

Select a feature model(Select one among these):
1. color_moments
2. hog
3. resnet50_layer3
4. resnet50_avgpool
5. resnet50_fc
6. resnet_softmax

Enter input: resnet50_avgpool
Enter k value: 5
Enter one of the following dimensional reduction techniques on the chosen feature model:

1. SVD
2. NNMF
3. LDA
4. k-means

Enter dimensionality reduction technique: 2
(101, 101)
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/sklearn/decomposition/_nmf.py:1710: ConvergenceWarning: M
aximum number of iterations 200 reached. Increase it to improve convergence.
  warnings.warn(
(101, 5)
Top 10 labels:

label id:52, category: inline_skate, distance: 0.0
label id:57, category: laptop, distance: 0.0019725675273313537
label id:46, category: grand_piano, distance: 0.004515687073725835
label id:5, category: Airplanes_Side_2, distance: 0.009582635886364953
label id:26, category: crab, distance: 0.015046149675863707
label id:48, category: headphone, distance: 0.01608552160437081
label id:92, category: trilobite, distance: 0.016753298057839122
label id:80, category: scissors, distance: 0.018024112300514553
label id:62, category: mayfly, distance: 0.021741418342740154
label id:94, category: watch, distance: 0.02175022261180183
```

Fig8.1: Image ID 0, LS3, K=10, AvgPool, NNMF, k=5

```
Enter query image ID: 8676
Enter any of the Latent Semantics:
1. LS1
2. LS2
3. LS3
4. LS4

Enter your choice number: 3
Enter K value for finding K similar labels: 10

Select a feature model(Select one among these):
1. color_moments
2. hog
3. resnet50_layer3
4. resnet50_avgpool
5. resnet50_fc
6. resnet_softmax

Enter input: resnet50_avgpool
Enter k value: 5
Enter one of the following dimensional reduction techniques on the chosen feature model:

1. SVD
2. NNMF
3. LDA
4. k-means

Enter dimensionality reduction technique: 2
(101, 101)
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/sklearn/decomposition/_nmf.py:1710: ConvergenceWarning: M
aximum number of iterations 200 reached. Increase it to improve convergence.
  warnings.warn(
(101, 5)
Top 10 labels:

label id:2, category: Leopards, distance: 0.0
label id:4, category: accordion, distance: 0.0
label id:21, category: cellphone, distance: 0.0
label id:85, category: stapler, distance: 0.0
label id:91, category: tick, distance: 0.0
label id:38, category: euphonium, distance: 0.01632507181672471
label id:3, category: Motorbikes_16, distance: 0.016909971845887932
label id:27, category: crayfish, distance: 0.017345519825894593
label id:17, category: camera, distance: 0.017536008682081935
label id:99, category: wrench, distance: 0.0177442307733505
```

Fig 8.2: Image ID 8676, LS3, K=10, AvgPool, NNMF, k=5

```
lalitarvind@Lalits-MacBook-Pro tasks % python3 task8.py
Enter '1' if you want to give an image ID as input or enter '2' if you want to give Image File as an Input:
Enter query type: 1
Enter query image ID: 0
Enter any of the Latent Semantics:
1. LS1
2. LS2
3. LS3
4. LS4

Enter your choice number: 4
Enter K value for finding K similar labels: 10

Select a feature model(Select one among these):
1. color_moments
2. hog
3. resnet50_layer3
4. resnet50_avgpool
5. resnet50_fc
6. resnet_softmax

Enter input: resnet50_layer3
Enter k value: 5
Enter one of the following dimensional reduction techniques on the chosen feature model:

1. SVD
2. NNMF
3. LDA
4. k-means

Enter dimensionality reduction technique: 3
iteration: 1 of max_iter: 10
iteration: 2 of max_iter: 10
iteration: 3 of max_iter: 10
iteration: 4 of max_iter: 10
iteration: 5 of max_iter: 10
iteration: 6 of max_iter: 10
iteration: 7 of max_iter: 10
iteration: 8 of max_iter: 10
iteration: 9 of max_iter: 10
iteration: 10 of max_iter: 10
[[0.20286439 0.20188958 0.19823007 0.19892419 0.19809178]
 [0.20288573 0.20143461 0.19829386 0.19813222 0.19925358]
 [0.20260743 0.20191672 0.19844372 0.19867081 0.19836132]
 ...
 [0.19821267 0.19987048 0.19981118 0.20218563 0.19992004]
 [0.19895032 0.20002543 0.20109645 0.20011426 0.19981354]
 [0.19815412 0.19908156 0.20088298 0.20153923 0.20034211]]
calculating js divergence
Top 10 labels:

label id:36, category: elephant, distance: 0.001941173966769381
label id:65, category: minaret, distance: 0.0021110403432054747
label id:13, category: brain, distance: 0.002123563554330093
label id:11, category: binocular, distance: 0.002423443132264473
label id:96, category: wheelchair, distance: 0.00248205574122762
label id:88, category: stop_sign, distance: 0.0025856919466657334
label id:8, category: barrel, distance: 0.002591060216203493
label id:23, category: chandelier, distance: 0.0025964901096654914
label id:25, category: cougar_face, distance: 0.002616956595272389
label id:12, category: bonsai, distance: 0.002661787162934311
```

Fig 8.3: Image ID 0, LS4, K=10, Layer3, LDA, k=5

```
lalitarvind@Lalits-MacBook-Pro tasks % python3 task8.py
Enter '1' if you want to give an image ID as input or enter '2' if you want to give Image File as an Input:
Enter query type: 1
Enter query image ID: 8676
Enter any of the Latent Semantics:
1. LS1
2. LS2
3. LS3
4. LS4

Enter your choice number: 4
Enter K value for finding K similar labels: 10

Select a feature model(Select one among these):
1. color_moments
2. hog
3. resnet50_layer3
4. resnet50_avgpool
5. resnet50_fc
6. resnet_softmax

Enter input: resnet50_layer3
Enter k value: 5
Enter one of the following dimensional reduction techniques on the chosen feature model:

1. SVD
2. NNMF
3. LDA
4. k-means

Enter dimensionality reduction technique: 3
iteration: 1 of max_iter: 10
iteration: 2 of max_iter: 10
iteration: 3 of max_iter: 10
iteration: 4 of max_iter: 10
iteration: 5 of max_iter: 10
iteration: 6 of max_iter: 10
iteration: 7 of max_iter: 10
iteration: 8 of max_iter: 10
iteration: 9 of max_iter: 10
iteration: 10 of max_iter: 10
[[0.20144228 0.1993636  0.20002759 0.19981231 0.19935422]
 [0.20213069 0.199495   0.199244   0.19971935 0.19941096]
 [0.20163461 0.19894245 0.199983   0.19948659 0.19995334]
 ...
 [0.19760761 0.19998964 0.19987437 0.20285498 0.19967339]
 [0.19857954 0.19994272 0.2005839  0.20013002 0.20076383]
 [0.19644251 0.20129486 0.20060702 0.20276557 0.19889004]]
calculating js divergence
Top 10 labels:

label id:84, category: soccer_ball, distance: 0.0005101080314569979
label id:25, category: cougar_face, distance: 0.0007240990439559169
label id:90, category: sunflower, distance: 0.0007757801748428509
label id:43, category: garfield, distance: 0.0009410972442243524
label id:100, category: yin_yang, distance: 0.0010496786954343042
label id:82, category: sea_horse, distance: 0.0010866989123032654
label id:55, category: ketch, distance: 0.001189382886508556
label id:1, category: Faces_3, distance: 0.0012892859399457552
label id:97, category: wild_cat, distance: 0.0012894184560108678
label id:61, category: mandolin, distance: 0.0012950607010520276
```

Fig 8.4: Image ID 8676, LS4, K=10, Layer3, LDA, k=5

Assumptions & Observations & Improvements:
We assume that the calculated representative image rightly describes the label both in latent space as well as the feature space.

Observations: It is evident that the assigned labels do not accurately reflect the expected labels in some test cases. Several factors could contribute to this discrepancy. Firstly, the representatives chosen to represent labels may not effectively capture the characteristics of both the feature space and the latent space, leading to misclassification. Secondly, outliers within each category may have inadvertently become the designated representatives, introducing label confusion. Additionally, a potential limitation is the insufficient number of samples per category, limiting the model's ability to learn distinctive patterns. Lastly, the number of latent features extracted might be inadequate, lacking the discriminative power needed to distinguish between the various categories.

Improvements: Optimizing parameters, such as adjusting the value of 'k,' can enhance the quality of results. Secondly, expanding the dataset to include a larger number of examples for training can provide more robust learning and better generalization. Additionally, experimenting with a variety of feature reduction techniques may uncover more effective approaches to capture essential information. Finally, the use of alternative distance measures in computing distances between data points can potentially refine the model's ability to discriminate between instances.

Task 9: Find K Closest Labels given a Query Label using Latent Semantics from Task 3-6

Query Input Given:
The program takes a label (l), a user-selected latent semantic, and a positive integer k as inputs.

Implementation:
Before running Task 9, representative images of all labels for all feature models are stored in a database. User is then asked what feature model and the program then selects the representative image for the given label under that feature model. It then compares the query label representative image in the selected latent space (for that feature model) with all other representative images. The top k similar labels (labels of those top k representative images) are printed along with their distance scores. If the latent space is LS3 (label-label matrix), the query label is directly compared with label-label latent semantics.

Design Choices & Decisions:
Representative Image for a label: When given a label, instead of iterating over all the images of that label to find the similarities with other labels of images, we choose to create a representative image for every label, so it will improve the code flow and the execution time.

Expected Outcomes:
The program outputs the k most likely matching labels along with their scores under the selected latent space.

Outputs:

```
Label : 0 , weight : 0.0
Label : 1 , weight : 3.268661673111796
Label : 83 , weight : 14.604260850370034
Label : 43 , weight : 15.449098016323306
Label : 88 , weight : 15.887352267620917
Label : 100 , weight : 16.09326463365298
Label : 32 , weight : 16.24267035790687
Label : 82 , weight : 16.794117704387777
Label : 48 , weight : 16.83994327838534
Label : 85 , weight : 16.845816417114314
```

Fig 9.1: Label 0, T3, Resnet, KMEANS

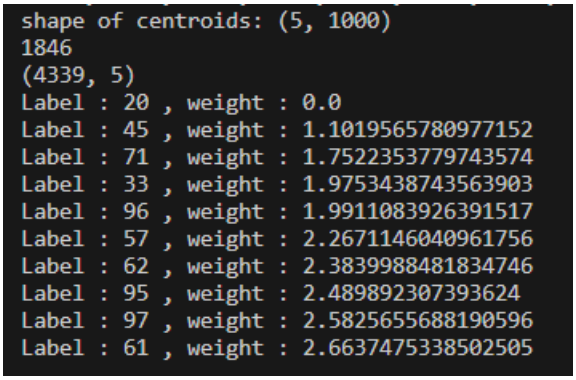

Fig 9.2: Label 20, T3, Resnet, KMEANS

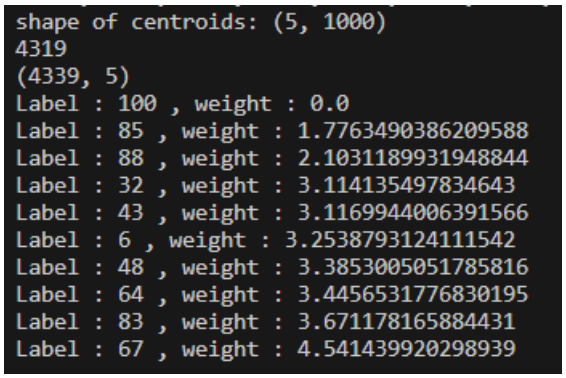

Fig 9.3: Label 100, T3, Resnet, KMEANS

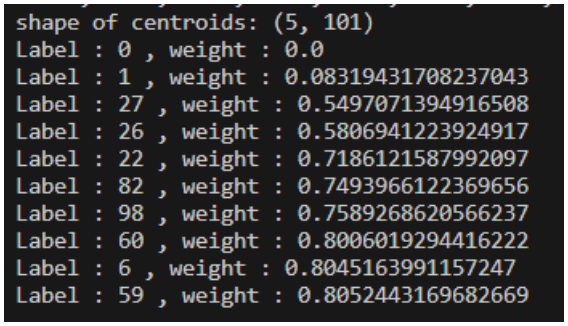

Fig 9.1: Label 0, T5, Resnet, KMEANS

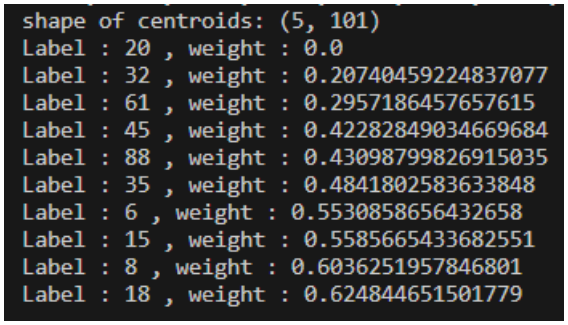

Fig 9.2: Label 20, T5, Resnet, KMEANS

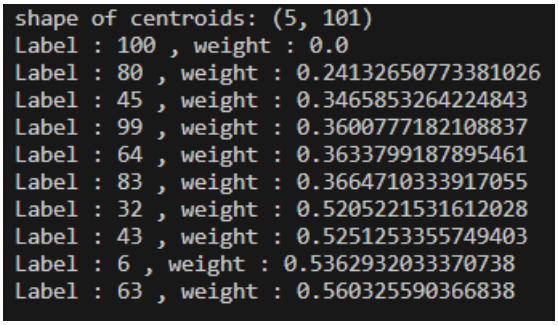

Fig 9.3: Label 100, T5, Resnet, KMEANS

Assumptions & Observations & Improvements:
We assume that representative images for all labels are stored in a database. All the images of the label are clustered and the image that is closer to the cluster centroid is our representative image of that label.

Task 10: Find K Similar Images given a Query Label using Latent Semantics from Task 3-6

Query Input Given:

Implementation:
Similar to Task 9, the program retrieves the representative image for the given label. It then compares the query image in the selected latent space with all other images in the latent space. The top k similar images are selected and printed along with their distance scores. If the latent space is LS3 (label-label matrix), the query label is directly compared with label-label latent semantics. The top k similar images (representative images of the labels) are printed along with their distance scores.

Design Choices & Decisions:
Representative Image for a label: When given a label, instead of iterating over all the images of that label to find the similarities with other labels of images, we choose to create a representative image for every label, so it will improve the code flow and the execution time.

Expected Outcomes:
The program visualizes the k most similar images along with their scores under the selected latent space.

Outputs:
All the query inputs take K=10 (number of similar images to be printed) and k=5 (latent space dimensions)

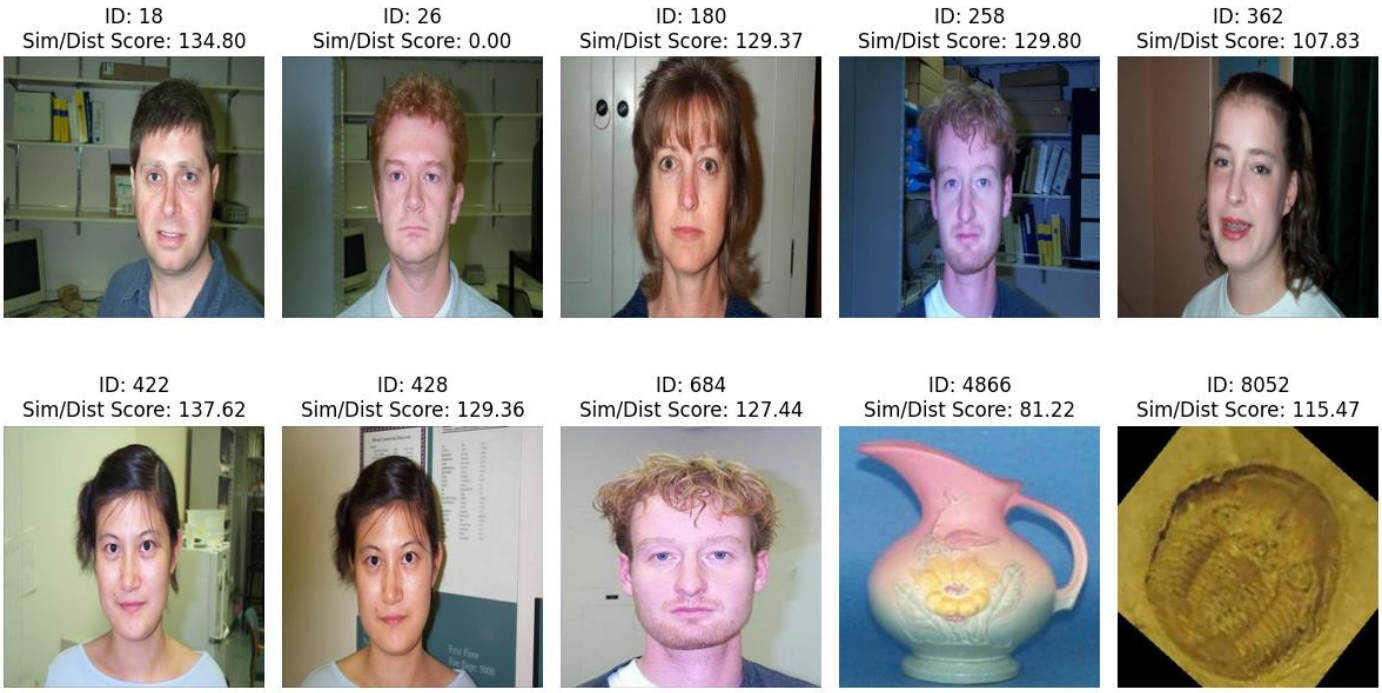

Fig 10.1: Label 0, T4, HOG

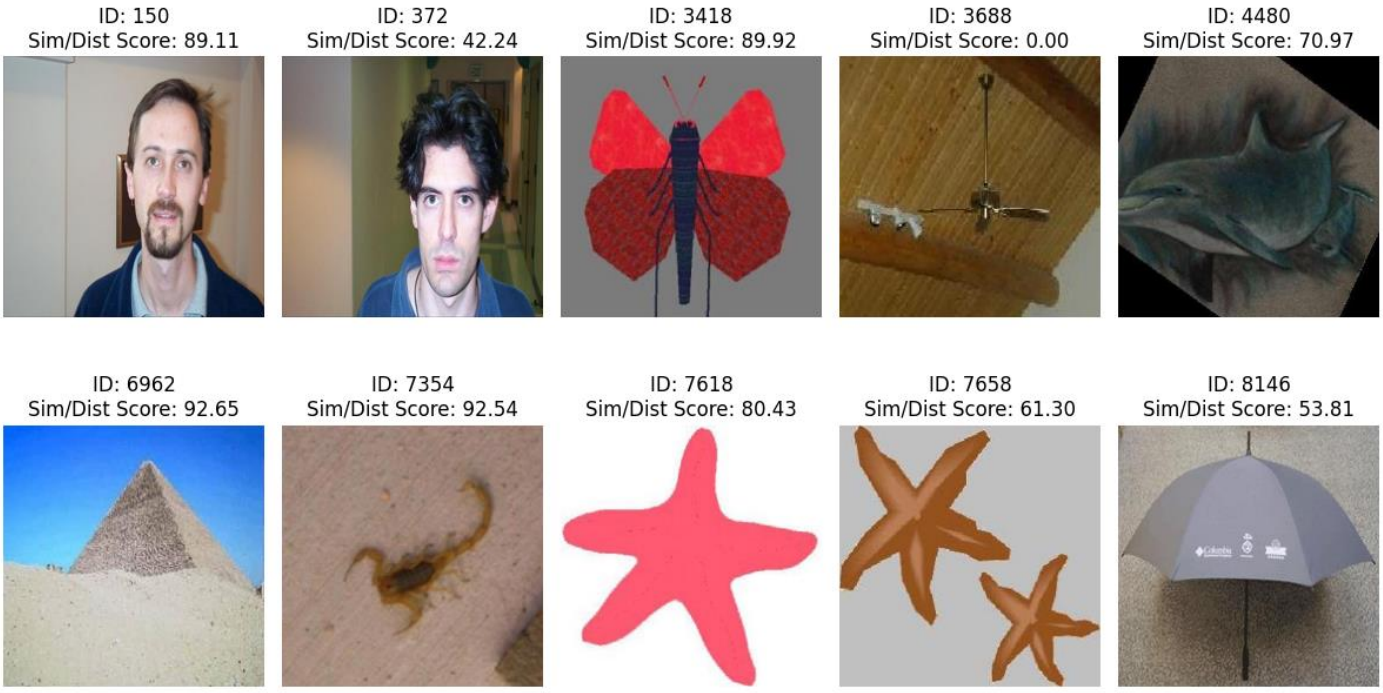

Fig 10.2: Label 20, T4, HOG

Fig 10.3: Label 100, T4, HOG

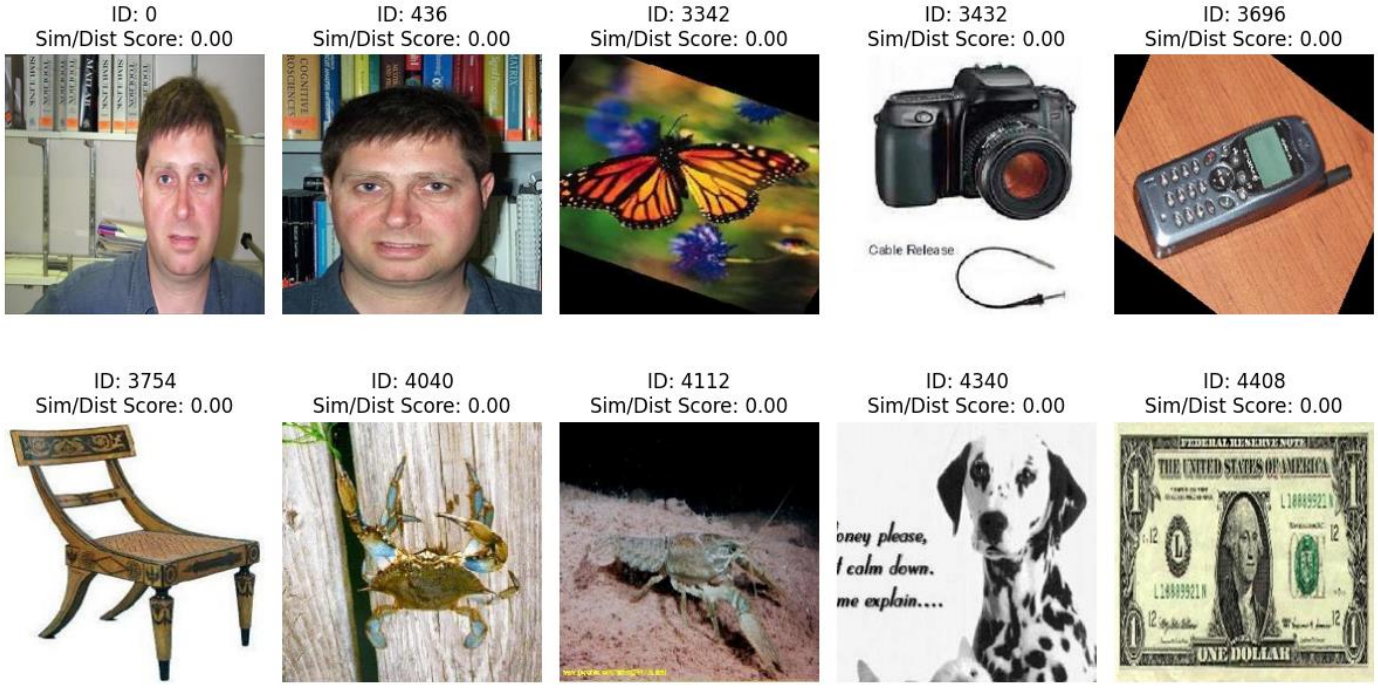

Fig 10.4: Label 0, T6, Resnet, SVD

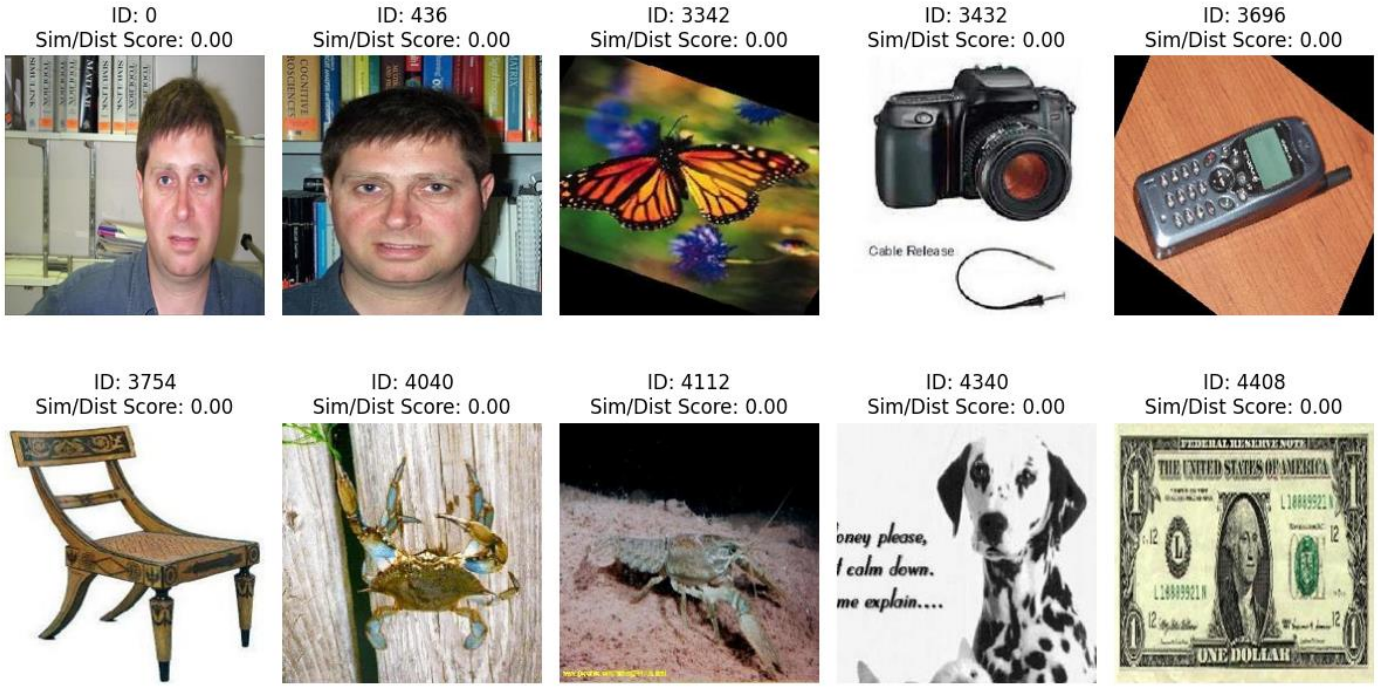

Fig 10.5: Label 20, T6, Resnet, SVD

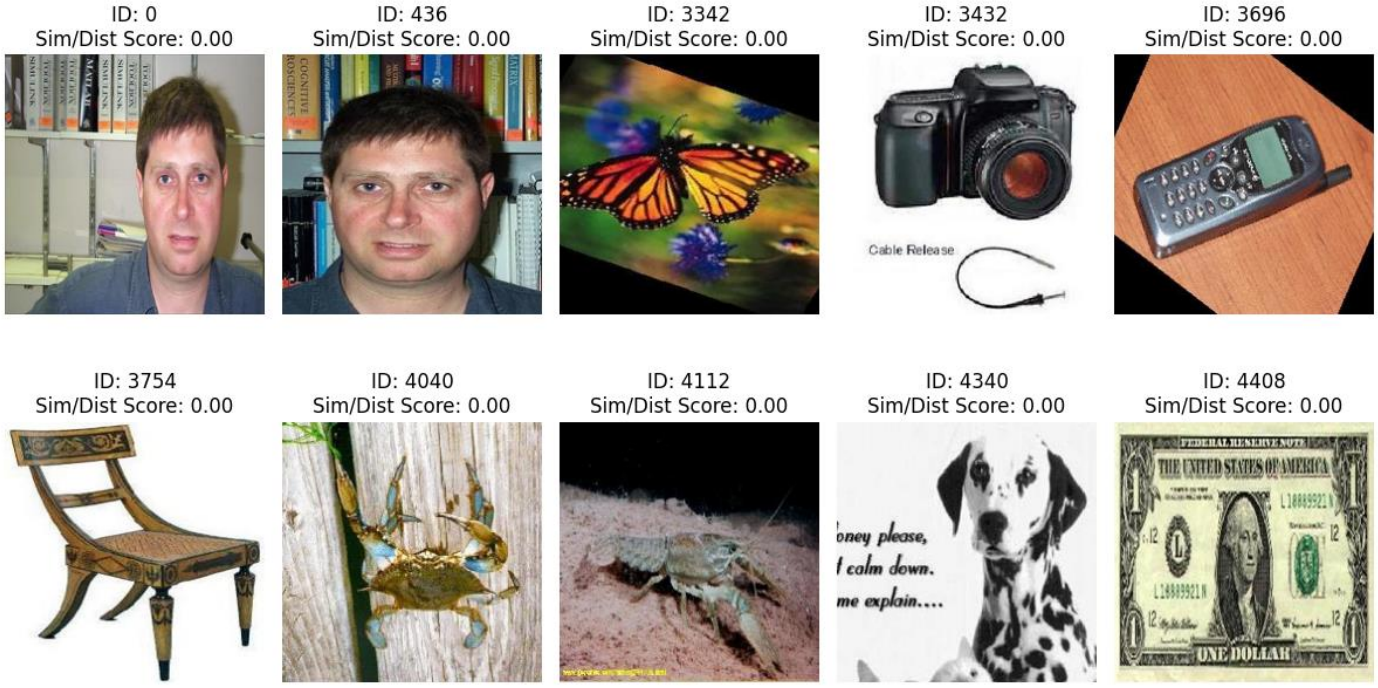

Fig 10.6: Label 100, T6, Resnet, SVD

Assumptions & Observations & Improvements:

In our approach, we make an assumption that representative images for each label are stored in a database. To determine the representative image for a label, we cluster all the images belonging to that label and select the image closest to the cluster centroid as the representative. However, our observations reveal that, in the case of the ResNet feature model, all the outputs are identical, and the distance score is consistently zero. This indicates the need for an improvement in the logic and methodology behind this latent semantic representation.

Task 11: Graph Representation of Images and Implementing Personalized PageRank Algorithm

Query Input Given:

Latent space, n, m, label l is given as inputs. Create a graph using the input query and perform latent space search which will be discussed in the implementation.

Implementation:

V represents the number of images in the database. (Vi, Vj) are the node pairs of images. Using the database, we find the cosine similarity between each image and find n-most similar images based on the similarity values. This similarity search is done in the given latent space. Now in the graph each image node is given n edges to its most similar nodes. With this we have created the graph G(V, E)

We then first construct a mapping of image IDs to their corresponding labels from the database collection. Then we define a personalized PageRank vector by assigning a value of 1 to nodes with the desired label (l) and 0 to all other nodes. The algorithm computes PageRank scores using NetworkX's PageRank function with a damping factor of 0.85 and the personalized vector. It sorts the nodes based on their PageRank scores in descending order and selects the top m nodes to return as the top m image IDs, which are multiplied by 2 to obtain the original image IDs in the database.

Design Choices & Decisions:

Networkx external library is used to create the graph and the same libraries PageRank algorithm is used to compute the PageRank vector. For T5, the labels representative images are used and visualized.

Expected Outcomes:

m similar images are identified and visualized accurately using the PageRank algorithm of graphs under the selected latent space.

Outputs:

All the given input queries are giving n=10, m=5 and k=5.

Fig 11.1: Label 0, FC

Fig 11.2: Label 20, FC

Fig 11.3: Label 55, FC

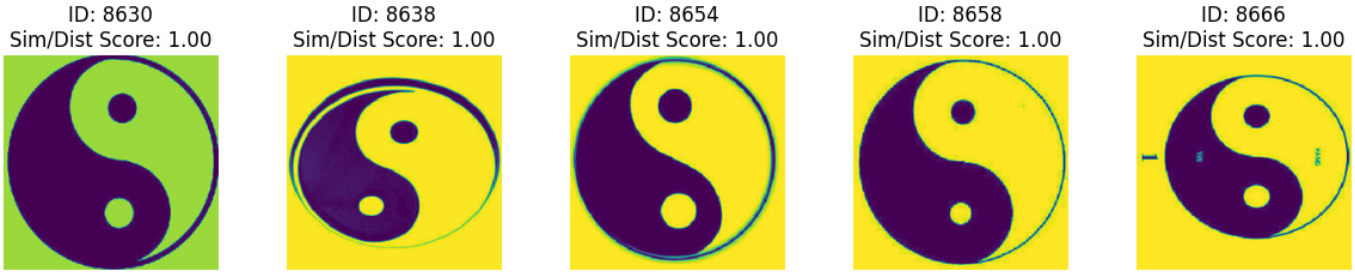

Fig 11.4: Label 100, FC

Fig 11.5: Label 0, T3, CM, SVD

Fig 11.6: Label 20, T3, CM, SVD

Fig 11.7: Label 55, T3, CM, SVD

Fig 11.8: Label 100, T3, CM, SVD

Assumptions & Observations & Improvements:

We assume that representative images for all labels are stored in a database. All the images of the label are clustered and the image that is closer to the cluster centroid is our representative image of that label. Using the PPR algorithm any of the input feature model or latent semantic gives accurate results. From this we can infer that representing images as nodes and edges as similarities among them, the latent space search gives more accurate results in identifying and visualizing the most similar m images.

System Requirements/Installation and Execution Instructions:

Before starting the phase2 execution (task0a.py, task0b.py, task1.py, task2a.py, task2b.py, task3.py, task4.py, task5.py, task6.py, task7.py, task8.py, task9.py, task10.py, task11.py), ensure that you have the necessary packages and prerequisites installed. Below are the steps to set up your environment:

Install Anaconda, and VS Code IDE:
Install Required Packages: Required packages that need to be installed are torch, torch vision, PyMongo NumPy, matplotlib, PIL, SK-learn and tensorly.
MongoDB and Compass GUI: Ensure you have MongoDB Compass GUI installed to interact with the MongoDB database.
Dataset Directory: Set the caltech101_directory variable in your code to the directory path where you have the Caltech101 dataset stored on your local machine.
Prerequisite Executions:

1. Create data matrices: Open the VSCode terminal or your preferred terminal application. Navigate to the directory containing create_data_matrix.py and execute the python script using the following command: python create_data_matrix.py. It creates all the data matrices (4339 * 900) for color_moments and hog, (4339 * 1024) and (4339 * 1000) for resnet50 in “data_matrices” folder.
2. Create representative images database*:* Open the VSCode terminal or your preferred terminal application. Navigate to the directory containing create_rep_images.py and execute the python script using the following command: python create_rep_images.py. It creates a collection with name “labelrepresentativeimages” in the database which contains the representative images for all the labels in all feature spaces.
3. Create label-label similarity matrices: Open the VSCode terminal or your preferred terminal application. Navigate to the directory containing create_label_label_sm.py and execute the python script using the following command: python create_label_label_sm.py. It creates label-label similarity matrices for all the feature spaces in the “label_label_sm_matrices” folder.
4. Create image-image similarity matrices: Open the VSCode terminal or your preferred terminal application. Navigate to the directory containing create_image_image_sm.py and execute the python script using the following command: python create_image_image_sm.py. It creates image-image similarity matrices for all the feature spaces in the “image_image_sm_matrices” folder.
5. Create the output folder: Create a directory called “outputs” for storing the latent semantics and weight pairs.

Run the Python Script: Open the VSCode terminal or your preferred terminal application. Navigate to the tasks directory and execute the python scripts from task0 - task11.
Interact with the System: Once the script is running, follow the prompts in the terminal to select and perform tasks related to image feature extraction and similarity analysis.

Conclusion:

In the previous phase of the Multimedia & Web Databases project, we laid a strong foundation by developing a comprehensive system for image feature extraction and similarity analysis. By harnessing a diverse set of visual models, including Color Moments, Histogram of Gradients (HOG), and the powerful ResNet50 neural network, we enabled efficient retrieval and visualization of feature descriptors for the images.

The initial phase of our project involved the extraction of feature vectors, which were notably substantial in size, such as 900 dimensions for Color Moments and 1024 dimensions for ResNet50's intermediate layer, with a final layer of 1000 dimensions. These features provided us with valuable insights into the content of the images and allowed us to create a robust framework for subsequent phases.

In the current phase, we have elevated our project to the next level by introducing the concept of Latent Semantics. These latent features are derived from the rich feature spaces we previously established, and we have applied various dimensionality reduction techniques, including Singular Value Decomposition (SVD),

Non-negative Matrix Factorization (NNMF), Linear Discriminant Analysis (LDA), and k-means clustering, to distil the most salient information.

This innovative approach has allowed us to capture the underlying structures and patterns within our data, reducing the dimensionality while preserving the essence of the images. As a result, we have created a more compact representation of the visual content, which has proven to be invaluable for tasks like image retrieval, label matching, and similarity analysis.

Furthermore, we have explored CP Decomposition, a powerful technique that extends our latent semantics into a three-modal tensor space, encompassing images, features, and labels. This multi-dimensional approach has enabled us to uncover deeper relationships between these entities, providing an enriched understanding of the data.

Our project culminated in the creation of latent spaces, both for labels and images, each offering valuable insights into the dataset's structure. By applying personalized PageRank measures and exploring graph-based analysis, we were able to identify the most significant images relative to a given label, further enhancing the relevance and context of the images in our database.

In conclusion, our project has evolved from basic feature extraction to the advanced realm of Latent Semantics, offering a holistic approach to image analysis and retrieval. This multifaceted system provides powerful tools for visual data exploration, paving the way for more effective multimedia and web database management. Through meticulous implementation and exploration of various techniques, we have not only achieved the project's objectives but also opened new doors to innovative research and applications in the field of multimedia and web databases.

Potential Future Work:

Distance Metric Enhancement: In future work, addressing the challenge observed with color moments' Euclidean distance metric is imperative. Exploring and selecting more appropriate distance or similarity measures tailored to color-based features should be a priority. This could involve experimenting with alternative metrics like Earth Movers' distance or statistical measures designed to capture nuances in color distribution. Such exploration can potentially lead to significantly improved results for color-based feature analysis.

Normalization and Dimensionality Reduction: An essential step is to eliminate the assumption of not normalizing feature descriptors. Normalization can standardize feature values across different images, enhancing the consistency and interpretability of distance measures. Additionally, the high dimensionality of feature descriptors (e.g., 900, 1000, 1024) presents challenges in terms of computational complexity and storage. Future work should delve into dimensionality reduction techniques, such as Principal Component Analysis (PCA) or t-Distributed Stochastic Neighbour Embedding (t-SNE). These methods can effectively reduce dimensionality while preserving essential information, streamlining analysis and visualization processes.

Hybrid Models: Develop hybrid models that combine the strengths of different dimensionality reduction techniques. For example, you can investigate how combining LDA and SVD or NNMF with t-SNE can enhance the interpretability and effectiveness of latent semantics.

Evaluation Metrics: Develop and implement robust evaluation metrics to assess the performance of our system. Consider conducting user studies to gather feedback and refine our system based on user preferences and requirements.

Scalability and Efficiency: Optimize our system for scalability and efficiency, especially when dealing with large datasets. This may involve distributed computing, cloud-based solutions, or hardware acceleration.

Real-World Applications: Apply our system to real-world applications in fields such as image classification, recommendation systems, and content tagging. This will help validate the practicality and effectiveness of our techniques.

Bibliography/References (APA Citing):

1. Huang, S., Li, X., Candan, K. S., & Sapino, M. L. (2016). Reducing seed noise in personalized PageRank. *Social Network Analysis and Mining*, *6*, 1-25.
2. Mudrova, M., & Procházka, A. (2005, November). Principal component analysis in image processing. In *Proceedings of the MATLAB technical computing conference, Prague*.
3. Duchenne, O., Joulin, A., & Ponce, J. (2011, November). A graph-matching kernel for object categorization. In *2011 International conference on computer vision* (pp. 1792-1799). IEEE.

Appendix(Contributions):

In our project, we ensured a fair distribution of tasks, with each team member contributing equally to their respective responsibilities.

Khadyothan Choudari Dasari:

1. Implemented SVD & NNMF for dimensionality reduction (Task 3)
2. Created python scripts to pre process and generate the required data matrices which also include label-label similarity and image-image similarity matrices (Tasks 5 & 6)
3. Integrated the code written by other team members for dimension reduction techniques with Task 5 & 6 for the successful working of the tasks.
4. Implemented Task 7 & Task 11

Chandra Kishore Reddy Gurram:

1. Implemented CP decomposition (Task 4)
2. Implemented tasks 9 & 10
3. Integrated the code written by other team members for dimensionality reduction techniques with Task 7, 9 & 10 for the successful working of the tasks.

Lalit Arvind & Rahul worked collaboratively on task2b, LDA, task8 and other distance metrics and integrated the code with other team members code for the successful working of the tasks.

Lalit Arvind Balaji:

1. Implemented the Task2b.
2. Implemented LDA for dimensionality reduction (Task 3)
3. Implemented Task 8 and worked on different distance metrics for LDA.

Rahul Rajaram:

1. Implemented the Task2b.
2. Implemented LDA for dimensionality reduction (Task 3)
3. Implemented Task 8 and worked on different distance metrics for LDA.

Rohit Bathi:

1. Collaborated with Bevin on the implementation of KMEANS for dimensionality reduction (Task 4).
2. Contributed in integrating tasks 0 to 2a, task 5,6.

Bevin Biju Thomos:

1. Collaborated with Rohith on the implementation of KMEANS for dimensionality reduction (Task 4).
2. Contributed in integrating tasks 0 to 2a, task 5,6.

Despite these task divisions, our team maintained an environment of active collaboration and problem-solving. We collectively addressed challenges, made important design decisions, and resolved complex issues as they arose during the project. This collaborative spirit fostered a deep understanding of the project's intricacies, ultimately contributing to the success of our overall project outcomes.