CSE515 Phase 2

Group 21: Ariana Bui, Maryam Cheema, Kasi Chikkala
Yash Dhamecha, Peter Kim, Rajat Pawar

Abstract

In Phase 2 of the CSE 515 Multimedia and Web Databases project, our group focused on image feature extraction, dimensionality reduction, and graph analysis. Building on Phase 1, we used the various feature extraction methods and neural network model to feature components like Color Moments, Histograms of Oriented Gradients (HOG), and RESNET50 layers. We applied SVD, NNMF, LDA, and k-means for latent semantics extraction, and performed CP decomposition on a three-modal tensor. We also created and analyzed label-label and image-image similarity matrices. Notably, we implemented Personalized PageRank for significant image identification. This project provides a comprehensive exploration of multimedia data analysis techniques, including identifying k most similar images or k most likely matching labels depending on user-selected latent semantics and feature models.

Keywords – feature model, SVD, NNMF, LDA, k-means, latent semantics, LDA decomposition, CP decomposition

Introduction

Expanding on the previous work done, there were several new project tasks completed as part of this phase. The same five different feature spaces were used on the Cal-tec data set to generate data vectors for the new tasks – color moments, histograms of oriented gradients (HOG), ResNet-AvgPool-1024, ResNet-Layer3-1024, and ResNet-FC-1000.

In the previous phase, our team implemented a program which, given an imageID, user selected feature space, and positive integer k , identified and visualized the most similar k images, along with their scores under the selected feature space. In the new phase, one of our goals was to implement a program which did the same using a given query label instead of an imageID. Another goal was to identify and list k most likely matching labels along with their scores given an imageID, positive integer k, and feature space or the RESNET50 neural network model.

Following these tasks, there were also several latent semantic related tasks. The goal of these included reporting the top-k latent semantics extracted under a selected feature space and reporting the top-k latent semantics extracted using SVD, NNMF, LDA, and k-means dimensionality reduction and CP-decomposition of a three modal (image-feature-label) tensor under a selected feature weight.

Assumptions:
Fost task 11, the assumption is that the similarity matrix file from one of the previous tasks will already be generated beforehand to be used in creating the similarity graph.

Description of Proposed Solution/Implementation

In most tasks, the process involves identifying similar images or labels, extracting latent semantics, and implementing techniques such as SVD, NNMF, LDA, and K-Means for dimensionality reduction. The following provides a general idea for each of the techniques utilized in dimensionality reduction. It's important to note that the same decomposition process is applied in Task 4, Task 5, Task 6, and Task 7.

1. **Singular Value Decomposition (SVD):** This technique decomposes the label similarity matrix into three matrices - U, Σ, and V^T. U represents the relationship between labels and latent semantics, Σ contains the singular values which indicate the importance of each latent semantic, and V^T characterizes the latent semantics and their relationship with the original features. By truncating Σ to retain the top k singular values and corresponding columns of U and V^T, we reduce the dimensionality.

$$M = U \cdot \Sigma \cdot V^T$$

where U contains eigenvectors of $M \cdot M^T$, Σ contains singular values, and V includes eigenvectors of $M^T \cdot M$.

Key elements:

- Covariance Matrices: Compute covariance matrices from the input feature_matrix.
- Eigenvalues and Eigenvectors: Obtain eigenvalues and eigenvectors for both covariance matrices.
- Sort Eigenvalues: Arrange eigenvalues in descending order to identify principal components.
- Transformation Matrices: Create transformation matrices u and v_transpose from sorted eigenvectors.
- Singular Value Matrix (Sigma): Generate sigma by taking the square root of eigenvalues.
- Dimension Reduction: Slice matrices to retain top k components, reducing dimensionality.

2. **Non-negative Matrix Factorization (NNMF):** NNMF approximates the label similarity matrix as the product of two non-negative matrices - W and H. W represents the weights of labels in the latent space, while H represents the latent features. Through iterative optimization, NNMF converges to a reduced representation.

$$M \approx W \cdot H$$

where W and H are non-negative matrices approximating M.

- Initialization: Set max_iter to 200, initialize matrices W and H with random values.
- Positive Feature Matrix: Ensure the feature matrix is positive by applying get_positive_feature_matrix.
- Iterations: For a specified number of iterations, update the matrices W and H using NMF update rules.
    - Update H: Calculate W^TA and W^TWH, then update H accordingly.
    - Update W: Calculate AH^T and WHH^T, then update W.
- Calculate Latent Features: Compute the latent features (image_to_latent_features) using the updated H and feature matrix.

3. **Latent Dirichlet Allocation (LDA):** LDA is a probabilistic model that interprets documents (labels) as mixtures of latent topics. Each topic is characterized by a distribution of words (features), and each document is a mixture of these topics. In this project, we apply a similar concept to labels and latent semantics. LDA transforms the label similarity matrix into a lower-dimensional space, revealing the underlying topics. We are using sklearn libraries to implement LDA.

- LDA Model Initialization: Create an LDA model with n_components=k, where k is the desired number of latent semantics.
- Model Fitting**:** Fit the LDA model on the positive feature matrix.
- Latent Features: Extract the latent feature-to-original feature matrix, denoted as latent_feature_to_original_feature = lda.components_.
- Calculate Latent Features: Compute the image-to-latent features using the dot product of the feature matrix and the transposed latent feature-to-original feature matrix (feature_matrix X latent_feature_to_original_feature^T).

4. **K-Means:** In this context, K-Means clustering is employed to group labels into *k* clusters. The centroids of these clusters represent the latent semantics, while each label is assigned to its closest centroid. Here we are following, iterative steps of assigning labels to clusters based on centroids and updating centroids based on label assignments. The specifics of these computations are algorithmic and not expressed by explicit formulas.
    - Initialization: Initialize the centroids randomly by selecting k data points from the feature matrix without replacement.
    - Number of Iterations: Set the maximum number of iterations to 100.
    - Iterative Process: For each iteration:
        - Assignment Step: Assign each data point to the nearest centroid using the cosine similarity metric.
        - Update Step: Recalculate the centroids by taking the mean of the data points assigned to each centroid.
    - Convergence Check: Verify if the centroids have converged. If they have not changed, the algorithm has converged.
    - Compute Latent Features: Calculate the latent features as the Euclidean distance between each data point and the centroids.

Below are some of the functionality which is common to most of the task:

1. **Libraries and Setup:** These programs utilize various Python libraries including `os`, `pickle`, `re`, `datetime`, `PIL`, `numpy`, `torch`, `torchvision`, `matplotlib`, and `pymongo`. These libraries provide functionalities for tasks like file operations, image processing, numerical computations, and database interactions.

2. **Preprocessing and Feature Extraction:**The program employs a pre-trained ResNet-50 model (`CNN_MODEL`) for feature extraction. It defines several functions (`preprocess_image`, `extract_feature_vector`, etc.) to process images before feeding them to the model and to extract features from specific layers. For the rest of the feature descriptors, we are reusing the features extracted from phase 1.

3. **Saving Latent Semantics**

    a. The resulting latent features serve as a compact and informative representation of label similarities. They are stored in a properly named output file for future use and analysis.
    b. We use the Python `pickle` library to save Latent Semantics. `Pickle` allows us to serialize and deserialize objects, making it easy to save and load data structures like dictionaries, lists, and arrays.
    c. The naming convention used a structured format that includes information about the task, feature model, dimensionality reduction technique, and the value of 'k' (the number of latent semantics).
    d. Example File Name:

        `Task#_FeatureModel_DimReductionTechnique_k.pkl`

        Example: `Task5_ResNet_Layer3_1024_SVD_10.pkl`

    e. We create a dictionary containing the latent semantics, usually with two main keys: `'Image_to_latent_features'`

        and `'latent_feature_to_original_feature'`.

    f. The `'image_to_latent_features'` key holds a matrix that maps images to their latent features.
    g. The `'latent_feature_to_original_feature'` key holds the transformation matrix that can be used to revert the latent features back to the original feature space if needed.
    h. The `pickle.dump()` function is used to write the data to a binary file.
    i. When we need to retrieve the latent semantics for further analysis, we use `pickle.load()` to deserialize the data from the file.

4. **Printing Label-Weight Pairs**

    After performing dimensionality reduction, we obtain a set of latent features, each representing a reduced-dimensional space. For each of these latent features, we associate labels with their respective weights, essentially quantifying the influence of each label on the feature. These label-weight pairs are then sorted in descending order based on the weight, preserving the index, so that we know which particular latent feature is most dominant. This ensures that labels with the highest influence are presented first, providing a clear indication of their significance.

5. **Loading Latent Semantics**

   The program executes the process of loading latent semantics by first constructing a file path using the chosen feature model, dimensionality reduction technique, and k value. This path is essential for locating and retrieving the relevant latent semantics. Subsequently, the program proceeds to load these latent semantics from the specified file, ensuring they are available for further processing.The naming convention for the latent semantics file follows the pattern:

   `Outputs/{Task}/{Feature Model}/{Dimensionality Reduction Technique}_{k}.pkl`

   When extracting the pickle file, several parameters are taken into account. These include the specific task number used to generate the pickle file, the chosen feature model (which could be HOG, CM, L3, AvgPool, FC, or RESNET), the selected dimensionality reduction technique (such as SVD, NNMF, LDA, or k-means), and the reduced dimensions denoted by kk. Once these parameters are determined, a file path is constructed accordingly. Subsequently, the program employs the pickle.load() function, a critical step in the process. This function plays a pivotal role in decoding the serialized data, effectively converting it back into a Python object. As a result, the latent semantics become readily accessible for further analysis. The loaded data is then stored in a variable, often as a dataframe for ease of manipulation. This systematic approach ensures that the relevant latent semantics are retrieved and available for subsequent processing.

0. Task 0: Feature Extraction and Similarity Visualization Using RESNET50-Based Feature Spaces

Task 0.a: In this task, we employed a pre-trained RESNET50 neural network model to process even-numbered labeled images from the Caltec101 dataset. These images were transformed into five unique feature spaces, namely Color Moments (CM10x10), Histograms of Oriented Gradients (HOG), ResNet-AvgPool-1024, ResNet-Layer3-1024, and ResNet-FC-1000. The resulting data vectors, together with the initial image labels, were meticulously stored in the database for further analysis and retrieval.

In this phase, we are essentially replicating the process from phase 1, with the distinction that we are exclusively working with even-numbered image IDs. While processing the feature descriptor, we just added a condition for the even odd check, if the number is even, we process that image else continue.

Task 0.b: This task is validation of task 0.a which is similar to task 3 of phase 1.

1. Task 1: Label-Based Image Retrieval Using Feature Spaces

Task 1 involves developing a program that takes three inputs: (a) a query label, denoted as "l," (b) a user-selected feature space, and (c) a positive integer "k." The goal is to identify and visualize the k most relevant images associated with the specified label, considering the chosen feature space. To achieve this, the program first retrieves the feature vectors of all images with the given label. Next, it calculates the similarity scores between the query label's feature vector and those of the other images. The k images with the highest similarity scores are then presented to the user along with their respective scores. This provides a clear visualization of the most relevant images based on the chosen feature space. The implementation utilizes appropriate data structures and algorithms to efficiently compute and display the results.

2. Task 2: Image-Based Label Prediction Using Feature Spaces & RESNET50 Model

Task 1 involves developing a program that takes three inputs: (a) a query label, denoted as "l," (b) a user-selected feature space, and (c) a positive integer "k." The goal is to identify and visualize the k most relevant images associated with the specified label, considering the chosen feature space.

- It begins by retrieving the feature vectors of all images associated with the specified label.
- Then, the program computes the mean of these extracted vectors to obtain a reference vector.
- This reference vector is subsequently utilized to identify the k most similar images.
- The program proceeds to calculate the similarity scores between the mean feature vector of the query label and those of the remaining images.
- Finally, it presents the k images with the highest similarity scores to the user, along with their respective scores. This visual representation offers a clear insight into the most relevant images within the chosen feature space.

We have used an in-built numpy mean function to get label mean vector, and for comparison we progressed with cosine similarity.

3. Task 3 (LS1): Top-k Latent Semantics Extraction and Storage

Task 3 (LS1) involves utilizing image-feature similarity matrix, applying dimensionality reduction techniques, and saving the resulting latent semantics. The main goal is to reduce the dimensionality of image features while retaining important information. We followed below implementation to tackle this task:

1. **Image feature Matrix:** We start by taking in input the image feature matrix from phase one and previous tasks. This matrix will be of size nxm which will be 4339xm. Depending on what the feature model is selected m will vary, for example, in case of FC m = 1000.

2. **Dimensionality Reduction:**
    a. This step condenses the image feature information while retaining essential characteristics. Four reduction techniques are offered, mentioned previously. Each technique applies a unique approach to reducing the dimensionality of the label space, mentioned before. This is executed in accordance with the previously outlined procedure, based on the decomposition technique called.
    b. For a given value of k and the chosen dimensionality reduction technique, the matrix will be decomposed into three components: $n \times k$, $k \times k$, and $k \times m$ matrices.
    c. For instance, if we take k to be 10 and employ the NNMF technique, we will obtain two decomposed matrices: W and H, with sizes of 4339×10 for W and 10×4339 for H.
    d. In the case of SVD, if we take k to be 10, we will obtain three decomposed matrices: U, Σ, and V^T, with sizes of 4339×10,10x10, and 10x4339 respectively. These matrices will represent the reduced feature space, singular values, and latent features.
    e. Unlike SVD and NNMF, LDA focuses on maximizing the separation between classes in the reduced feature space. It achieves this by finding linear combinations of the original features that best separate the classes. The resulting matrices will be 4339×10 and 10x4339, representing the transformed data and the original-to-reduced feature mapping.
    f. The output of K-means will be 10 centroids in the feature space, each with a dimensionality of 4339. This means we obtain a 10×4339 matrix representing the centroids in the original feature space.
3. **Saving Latent Semantics:** The resulting latent features serve as a compact and informative representation of label similarities. They are stored in a properly named output file for future use and analysis. This is executed in accordance with the previously outlined procedure.
4. **Printing Label-Weight Pairs:** To facilitate interpretation, the latent features are printed in the form of label-weight pairs. These pairs are presented in descending order of weight, providing insight into the most significant label relationships. This is executed in accordance with the previously outlined procedure

4. Task 4 (LS2): Top-k Latent Semantics Extraction using CP-decomposition

To perform CP-decomposition of a three modal (image-feature-label) tensor under the selected features space, we first needed to create the tensor. The tensor is structured as follows.

[number of Images] x [Feature Length] x [number of Labels]

In the case of this project, the tensor has the dimensions (4338 x 900 x 101). This tensor is passed through the parafac function from tensorly.decomposition. The resulting decomposition results in weights and factor matrices.

[number of images] x [rank]
[feature length] x [rank]
[number of labels] x [rank]

There are three factor matrices, one for each mode. Each matrix has *k* columns, each representing a component in the decomposition. The weights from the parafac function indicate the importance of each component in the decomposition. Each row represents the contribution of that component to the original tensor.

5. Task 5 (LS3) Generating Latent Semantics from Label Similarities

Task 5 (LS3) involves creating a label-label similarity matrix, applying dimensionality reduction techniques, and saving the resulting latent semantics. The main goal is to reduce the dimensionality of label similarities while retaining important information. We followed below implementation to tackle this task:

1. **Label Similarity Matrix Creation:** We start by generating a square matrix representing label similarities. This is done using a selected feature model to compute similarities between different labels. For each label, a mean vector of feature data is calculated, forming the basis for comparison. The matrix is populated with cosine similarity scores, capturing the relationships between labels.

$$Cosine\ Similarity(A, B) = A \cdot B / \|A\| \cdot \|B\|$$

Here:

- $A \cdot B$ represents the dot product of vectors A and B.
- $\| A \|$ and $\| B \|$ denote the Euclidean norms (lengths) of vectors A and B, respectively.

This formula yields a value between -1 and 1. As we have a total of 101 labels (0-100), we will get a matrix of size 101x101.

2. **Dimensionality Reduction:**
    - This step condenses the label similarity information while retaining essential characteristics. Four reduction techniques are offered, mentioned previously. Each technique applies a unique approach to reducing the dimensionality of the label

space, mentioned before. This is executed in accordance with the previously outlined procedure, based on the decomposition technique called.

Note: In this scenario, visualizing the *m* vector is challenging due to the use of square matrices. Had we been working with a feature descriptor matrix, we could easily visualize it. This is because in that case, the number of features would not be equal to the number of labels, providing a more intuitive understanding

- For a given value of *k* and the chosen dimensionality reduction technique, the matrix will be decomposed into three components: $n \times k$, $k \times k$, and $k \times m$ matrices.
- For instance, if we take *k* to be 10 and employ the NNMF technique, we will obtain two decomposed matrices: *W* and *H*, with sizes of 101×10 for *W* and 10×101 for *H*.
- In the case of SVD, if we take *k* to be 10, we will obtain three decomposed matrices: U, Σ, and V^T, with sizes of 101×10,10x10, and 10x101 respectively. These matrices will represent the reduced feature space, singular values, and latent features.
- Unlike SVD and NNMF, LDA focuses on maximizing the separation between classes in the reduced feature space. It achieves this by finding linear combinations of the original features that best separate the classes. The resulting matrices will be 101×10 and 10x101, representing the transformed data and the original-to-reduced feature mapping.
- The output of K-means will be *10* centroids in the feature space, each with a dimensionality of 101. This means we obtain a 10×101 matrix representing the centroids in the original feature space.

3. **Saving Latent Semantics:** The resulting latent features serve as a compact and informative representation of label similarities. They are stored in a properly named output file for future use and analysis. This is executed in accordance with the previously outlined procedure.
4. **Printing Label-Weight Pairs:** To facilitate interpretation, the latent features are printed in the form of label-weight pairs. These pairs are presented in descending order of weight, providing insight into the most significant label relationships. This is executed in accordance with the previously outlined procedure.

6. Task 6 (LS4) Generating Latent Semantics from Image Similarities

Task 6 (LS4) involves creating an image-image similarity matrix, applying dimensionality reduction techniques, and saving the resulting latent semantics. The main goal is to reduce the dimensionality of image similarities while retaining important information. We followed below implementation to tackle this task:

1. **Image Similarity Matrix Creation:** We start by generating a square matrix representing Image similarities. This is done using a selected feature model to compute similarities between different images. For each image, its respective feature vector is calculated, forming the basis for comparison. The matrix is populated with cosine similarity scores, capturing the relationships between labels.

$$Cosine\ Similarity(A, B) = A \cdot B / \|A\| \cdot \|B\|$$

Here:

- $A \cdot B$ represents the dot product of vectors A and B.
- $\| A \|$ and $\| B \|$ denote the Euclidean norms (lengths) of vectors A and B, respectively.

This formula yields a value between -1 and 1. As we have a total of 4339 images (0-4338), we will get a matrix of size 4339x4339.

NOTE: As mentioned in task 1, we are not using the whole dataset, but we are just storing data for even indexed images.

2. **Dimensionality Reduction:**
    - This step condenses the image similarity information while retaining essential characteristics. Four reduction techniques are offered, mentioned previously. Each technique applies a unique approach to reducing the dimensionality of the label space, mentioned before.
    - For a given value of k and the chosen dimensionality reduction technique, the image similarity matrix will be generated, resulting in 4339×4339 dimensions.
    - For instance, if we set k to be 10 and utilize the NNMF technique, we will create two decomposed matrices: W and H, with dimensions of 4339×10 for WW and 10×4339 for H.
    - In the case of SVD, with k set to 10, three decomposed matrices are produced: U, Σ, and V^T, with dimensions 4339×10, 10×10, and 10×4339 respectively. These matrices represent the reduced feature space, singular values, and latent features.
    - Contrary to SVD and NNMF, LDA aims to maximize class separation in the reduced feature space. It accomplishes this by discovering linear combinations of the original features that best segregate the classes. The resulting matrices will be 4339×10 and 10×4339, representing the transformed data and the original-to-reduced feature mapping.
    - In the case of K-means, the output will be 10 centroids in the feature space, each with a dimensionality of 4339. This leads to a 10×4339 matrix representing the centroids in the original feature space.
3. **Saving Latent Semantics:** The resulting latent features serve as a compact and informative representation of label similarities. They are stored in a properly named output file for future use and analysis.

4. **Printing Image-Weight Pairs:** To facilitate interpretation, the latent features are printed in the form of Image-weight pairs. These pairs are presented in descending order of weight, providing insight into the most significant label relationships.

7. Task 7: Image Retrieval Using Latent Semantic Representations

This task involves implementing a program that identifies and displays the top k most similar images to a given input image. The program supports various feature extraction techniques including HOG, Color Moments, and features from different layers of a ResNet-50 model. It also allows for dimensionality reduction using methods like SVD, NNMF, LDA, and K-means. The user interacts by providing an image ID or file name, selecting a latent semantic configuration, and specifying the number of similar images to retrieve (k). The program processes the input, computes feature vectors, and determines similarity scores to present the most relevant images. Additionally, it includes functionalities for input validation, result visualization, and output saving, providing a versatile solution for exploring image similarity in different latent semantic spaces.

1. Loading Latent Semantics: The program constructs a file path based on the selected feature model, dimensionality reduction technique, and value of k to retrieve the corresponding latent semantics and the task which was used to create this latent semantics. It then loads these latent semantics from the saved file, as mentioned above. This will be of size $n \times k$.
2. The program initially computes the feature representation based on the chosen feature model for a given image. The size of this feature varies depending on the selected model; for instance, for ResNet's fully connected layer, the feature descriptor is of size 1000.
3. Next, utilizing the stored reduced latent space, the program maps this feature descriptor to the corresponding reduced latent space. If 'k' was chosen to be 10 during the computation of latent features, the latent space will be of size 4339x10. This is achieved by projecting the feature vector into the latent space.
4. Subsequently, we obtain a matrix sized 4339x10 and a target feature descriptor of size 1x10. Our aim is to compare these in order to retrieve the top 'k' results based on cosine similarity.
5. The results are displayed using matplotlib.

8. Task 8: Label Matching Using Latent Semantic Representations

This task involves implementing a program that identifies and displays the top k most similar labels to a given input image. The program supports various feature extraction techniques including HOG, Color Moments, and features from different layers of a

ResNet-50 model. It also allows for dimensionality reduction using methods like SVD, NNMF, LDA, and K-means. The user interacts by providing an input image, selecting a latent semantic configuration, and specifying the number of similar labels to retrieve (k). The program processes the input, computes feature vectors, and determines similarity scores to present the most relevant images. Additionally, it includes functionalities for input validation, result visualization, providing a versatile solution for exploring label similarity in different latent semantic spaces.

1. Loading Latent Semantics: The program constructs a file path based on the selected feature model, dimensionality reduction technique, and value of k to retrieve the corresponding latent semantics and the task which was used to create this latent semantics. It then loads these latent semantics from the saved file, as mentioned above. This will be of size *n x k* that is *4339x10*.
2. For the reduced latent space of size 4339x10, obtained with k set to 10, and considering the 4339 even-numbered images, the program iteratively identifies the top images until the desired number of unique labels are retrieved. The count starts with k and decreases for each unique label obtained.

9. Task 9: Label Matching Using Latent Semantics

Task 9 involves creating a program that, given a label, user-selected latent semantics, and a positive integer k, identifies and lists the k most likely matching labels along with their scores under the selected latent space.

The program initiates by soliciting specific user inputs. These encompass the selection of a feature model from a provided set, choosing a dimensionality reduction method such as SVD, NNMF, LDA, or k-means, specifying a positive integer (k), designating a target label, and indicating the corresponding task number (e.g., 3, 4, 5, or 6). These inputs serve as crucial parameters for subsequent operations within the program. Next we need to perform:

1. Loading Latent Semantics: The program constructs a file path based on the selected feature model, dimensionality reduction technique, and value of k to retrieve the corresponding latent semantics and the task which was used to create this latent semantics. It then loads these latent semantics from the saved file, as mentioned above. This will be of size *n x k*.
2. Finding Top Matching Labels: The program calculates the similarity scores between the latent semantics from the extracted reduced feature space of the target label(1xk) and all other labels in the dataset. It sorts these labels based on their similarity scores, identifying the top k matching labels. So we would have a total of n similarities to choose from.
3. Displaying Results: The program presents the top matching labels along with their similarity scores to the user.

10. Task 10: Relevant Image Retrieval in Latent Space

Task 10 involves creating a program that, given a label, user-selected latent semantics, and a positive integer k, identifies and lists the k most relevant images along with their scores under the selected latent space.

The program initiates by soliciting specific user inputs. The inputs from the user include feature model from a provided set, choosing a dimensionality reduction method such as SVD, NNMF, LDA, or k-means, specifying a positive integer (k), designating a target label, and indicating the corresponding task number (e.g., 3, 4, 5, or 6). Next we perform the following steps:

1. Loading Latent Semantics: The program constructs a file path based on the selected feature model, dimensionality reduction technique, and value of k to retrieve the corresponding latent semantics and the task which was used to create this latent semantics. It then loads these latent semantics from the saved file, as mentioned above.This will be of size *n x k* which will be *4339x10*.
2. Subsequently, it loads the original feature descriptors for each image and associates labels from this collection with the reduced dimensions. It computes the mean factor for each label within the reduced space, resulting in a set of 101 mean vectors, each corresponding to a distinct label. This will leave us with a 101x10 matrix of mean vectors
3. Finding Top Matching Images: The program calculates the similarity scores between the label mean vector and the image latent vector. It sorts these images based on their similarity scores, identifying the top k relevant images.
4. Displaying Results: The program presents the top relevant images along with their similarity scores to the user using matplotlib python library.

11. Task 11: Page rank & personalized page rank

Task 11 involves accepting a feature model or a latent space, values of n and m and a label l. Value n corresponds to the set of n images that are similar to a specific image. Value m corresponds to m most significant images for a label l. We followed below implementation to tackle this task:

Constructing a similarity graph:

Using a similarity matrix created from previous tasks, we constructed a similarity graph G(V, E) where V corresponds to images and E contains node pairs vi, vj representing n most similar images in the given space.

Constructing an adjacency matrix:

Using the graph, we created an adjacency matrix for each of the images. The adjacency matrix contained a value of 1 if the images were connected in the graph and contained a value of zero if the images were not similar to each other.

Using Personalized PageRank Algorithm

Using the adjacency matrix, a probability score was calculated for each of the nodes (images) in the graph. As our initial seed value, a random index was generated. The probabilities were calculated based on whether the node was a neighbor of the node that was selected as the seed value. The probabilities were then sorted in descending order. Then the most significant images for a label are printed out.

Interface Specifications

The directory hierarchy adheres to the following structure:

```
Phase 2/
|
├── Code/
|   ├── task0.py~task11.py/
|   └── methods.py
|
├── Outputs/
|   ├── T0~T2
|   |     └── id_k.png ← image output stored here
|   |     ├── .txt ← label list stored here
|   ├── T3~T6/
|   |     ├── HOG~RESNET/
|   |     |     └── SVD~K-Means.pkl ← Latent semantics stored
|   |     └── Similarity matrix ← similarity stored
|
├── README.md
```

```
└── requirements.txt
```

0. Task 0: Feature Extraction and Similarity Visualization Using RESNET50-Based Feature Spaces

- Task 0a: This task is exactly similar to that of task 2 for phase 1.
    - For this task no inputs are required as the program will calculate features on its own.
- Task 0b:The user will be prompted for three inputs:
    - Image_ID or Image file
    - Feature Model
    - Positive integer *k*

- After the user fills in the appropriate values, the top-*k* images are identified using the selected feature space. The images along with their scores are visualized in the output file [insert output file format].

1. Task 1: Label-Based Image Retrieval Using Feature Spaces

Task 1 involves developing a program that takes three inputs: (a) a query label, denoted as "l," (b) a user-selected feature space, and (c) a positive integer "k."

2. Task 2: Image-Based Label Prediction Using Feature Spaces & RESNET50 Model

Task 2a:
The user will be prompted for three inputs:

- Image_ID or Image file
- Feature Model
- Positive integer *k*

Using the inputs, the program will use the selected feature space to identify the *k* most likely matching labels and their scores for the image given. These outputs are saved under the file [insert output file format here].

Task 2b:
The user will be prompted for two inputs:

- Image_ID or Image file
- Positive integer *k*

Using the inputs, the program will use the RESNET50 neural network model to find the *k* most likely matching labels and their scores for the image given. These outputs are saved under the file [insert output file format here]

3. Task 3 (LS1): Top-k Latent Semantics Extraction and Storage

The user will be prompted for three inputs:

- Feature Model
- Positive integer *k*
- Dimensionality Reduction Technique (SVD, NNMF, LDA, k-means)

After the user fills in the appropriate values, the top-*k* latent semantics are extracted using the specified dimensionality reduction technique under the specified feature space.

The outputs are stored into a file. The file is organized as a list of imageID-weight pairs, ordered in decreasing order of weights.

4. Task 4 (LS2): Top-k Latent Semantics Extraction using CP-decomposition

- The user will be prompted for two inputs:
    - Feature model
    - Positive integer *k*
- The user needs to select an appropriate feature model ("CM", "HOG", "AvgPool", "L3", "FC", "RESNET"), or they will receive an error message “Invalid feature model selection. Please select from the following list: CM, HOG, AvgPool, L3, FC, RESNET.” After the user enters their inputs, the top-*k* latent semantics are extracted using CP-decomposition.
- The outputs are stored into a file with the top *k* latent semantics. The semantics are presented in the form of a list of label-weight pairs, ordered in decreasing order of weights.

5. Task 5 (LS3) Generating Latent Semantics from Label Similarities

- Input:
    - Feature Model: example – ResNet-Layer3-1024
    - Value of k(number of latent features): example – 10
    - Dimensionality Reduction Technique: example – SVD

- Output:
    - Latent Semantics saved in the file: 'ResNet-Layer3-1024_SVD_10.pkl'
    - List of label-weight pairs in descending order of weights.

  The Label-Label Similarity Matrix is an (n x n) matrix, where 'n' represents the number of labels, resulting in a matrix size of 101x101 in this case. Upon processing this input, we obtain a set of Latent Semantics with dimensions (n x k), where 'n' refers to the number of labels, and 'k' is the specified value.

  Label-weight will be according to the most dominant latent semantic first.

  Please note that the output file will store the latent semantics for the respective dimensionality reduction technique within their respective folders. Additionally, the corresponding label-weight pairs will be displayed on the terminal.

6. Task 6 (LS4) Generating Latent Semantics from Image Similarities

- Input:
    - Feature Model: example – ResNet-Layer3-1024
    - Value of k(number of latent features): example – 10
    - Dimensionality Reduction Technique: example – SVD
- Output:
    - Latent Semantics saved in the file:
    - List of image-weight pairs in descending order of weights.

  The Image-Image Similarity Matrix is an (n x n) matrix, where 'n' represents the number of images, resulting in a matrix size of 4339x4339 in this case. Upon processing this input, we obtain a set of Latent Semantics with dimensions (n x k), where 'n' refers to the number of labels, and 'k' is the specified value, which would be 4339x10, where k = 10.

  Image-weight will be according to the most dominant latent semantic first.

  Please note that the output file will store the latent semantics for the respective dimensionality reduction technique within their respective folders. Additionally, the corresponding label-weight pairs will be displayed on the terminal.

7. Task 7: Image Retrieval Using Image Latent Semantic Representations

- Input:
    - Image input (file/id): 0
    - Take number: Task 3
    - Feature Model: ResNet-Layer3-1024
    - Value of k(number of images): 10
    - Dimensionality Reduction Technique: SVD
- Output:
    - Visualization of top k images
    - Image similarity metric of the images.

  The input feature is represented as a 1xm vector, which is further reduced for comparison in order to retrieve k images.

8. Task 8: Image Matching Using Label Latent Semantic Representations

- Input:
    - Image input (file/id): 0
    - Take number: Task 3
    - Feature Model: ResNet-Layer3-1024
    - Value of k(number of images): 10
    - Dimensionality Reduction Technique: SVD
- Output:
    - List of top k labels
    - Label similarity metric of the images.

  The input will be loaded from a file, consisting of latent features extracted from the Image-image similarity matrix and label-label similarity matrix. The program will output the top k labels based on either Euclidean distance or cosine similarity.

9. Task 9: Label Matching Using Label Latent Semantics

- Input:
    - Image input (file/id): 0
    - Take number: Task 3
    - Feature Model: ResNet-Layer3-1024
    - Value of k(number of images): 10
    - Dimensionality Reduction Technique: SVD
- Output:

- List of top k labels
- Label similarity metric of the label.

The input will be loaded from a file, consisting of latent features extracted from the label-label similarity matrix. The program will output the top k labels based on either Euclidean distance or cosine similarity.

10. Task 10: Image Matching Using Label Latent Semantics

- Input:
  - Input label l
  - Feature Model
  - Value of k(number of relevant images)
  - Dimensionality Reduction Technique
  - Task Number (Task 3, 4, 5 , 6)
- Output:
  - List of top k most relevant images along with the similarity scores

11. Task 11: Page rank & personalized page rank

- Input:
  - User will be prompted to choose a feature model
  - User will be prompted to input a value for m, n and the label
  - User will be prompted to choose if they want to use a feature model or a latent semantic model
  - If the user selects the latent semantics, they will be prompted for the dimensionality reduction technique, k value and a task number
- Output:
  - Top m relevant images for the label l
  - Node pairs vi , vj such that, for each subject vi , vj is one of the n most similar images in the database in the given space

System Requirements/Installation and Execution Instructions

System Requirements:

- Python (>= 3.6)
- MongoDB

- Required Python libraries (`pickle`, `re`, `datetime`, `PIL`, `numpy`, `torch`, `tqdm`, `torchvision`, `matplotlib`, `tensorflow`, and `pymongo`)

Installation:

1. **Library Installation:**
   - Open a command prompt or terminal window.
   - Use pip (Python's package installer) to install the required libraries:

```
pip install numpy pandas scikit-learn tqdm
```

Execution Instructions:

- ### Task 0: Feature Extraction and Similarity Visualization Using RESNET50-Based Feature Spaces

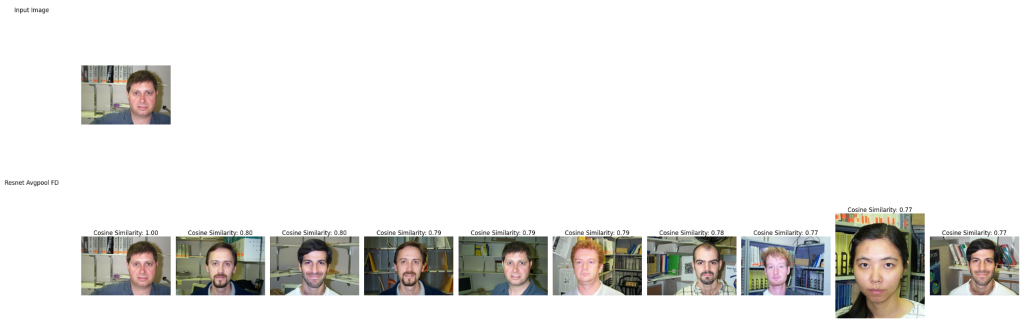
  - Navigate to the directory containing the Python scripts for Task 0.a & Task 0.b, which is present according to the directory structure provided previously.
  - Execute the program using a Python interpreter. For example:

```
python task0a.py
python task0b.py
```

```
Enter image ID between 0-8676 or image path.
0
Please pick one of the below options
1. HOG
2. Color Moments
3. Resnet Layer 3
4. Resnet Avgpool
5. Resnet FC
1
Select K to find K similar images to given input image
5
```

  - The program will process the data, extract features, and save the results in the mongo database.
  - It prompts the user for an (even or odd numbered) imageID or an image file, along with a user-selected feature space, and a positive integer k.

- The program generates features specific to the chosen category and calculates the similarity between the extracted feature vector and all other vectors stored in the database.
- It then retrieves the top k minimum similarities.
- Finally, it presents the output in the following format:



• Task 1: Label-Based Image Retrieval Using Feature Spaces

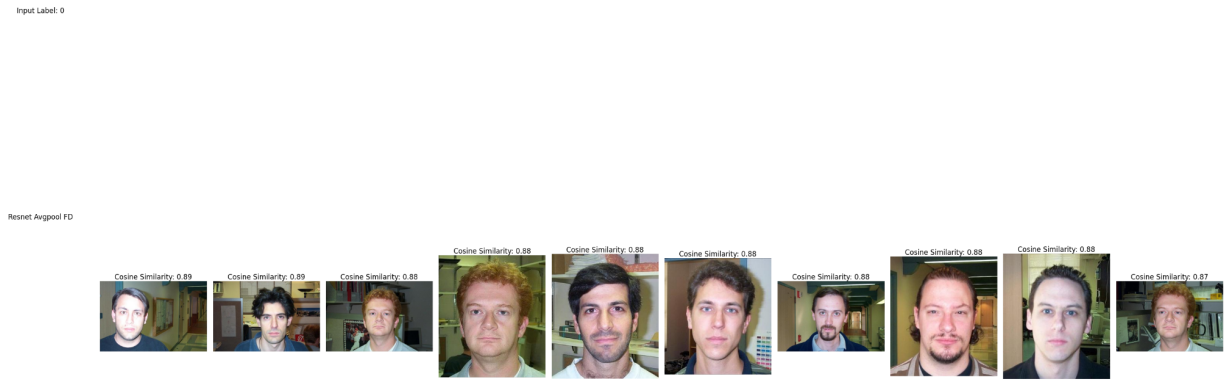
- Navigate to the directory containing the Python scripts for Task 1, which is present according to the directory structure provided previously.
- Execute the program using a Python interpreter. For example:

```
python task1.py
```

- It prompts the user for a label, along with a user-selected feature space, and a positive integer k.

```
Please enter image label
0
Please pick one of the below options
1. HOG
2. Color Moments
3. Resnet Layer 3
4. Resnet Avgpool
5. Resnet FC
1
Select K to find K similar images to given input image
4
```

- The program generates features specific to the chosen category and calculates the similarity between the extracted feature vector and all other vectors stored in the database.
- It then retrieves the top k minimum similarities.
- Finally, it presents the output in the following format:



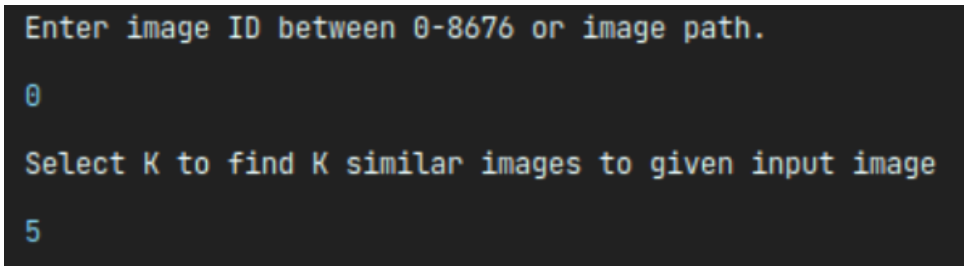
• Task 2: Image-Based Label Prediction Using Feature Spaces & RESNET50 Model

- Navigate to the directory containing the Python scripts for Task 2, which is present according to the directory structure provided previously.
- Execute the program using a Python interpreter. For example:

```
python task2a.py
python task2b.py
```

- It prompts the user for an image id or image file, along with a user-selected feature space, and a positive integer k for task 2a.
- It prompts the user for an image id or image file, and a positive integer k for task 2a.

```
Enter image ID between 0-8676 or image path.
0
Please pick one of the below options
1. HOG
2. Color Moments
3. Resnet Layer 3
4. Resnet Avgpool
5. Resnet FC
1
Select K to find K similar images to given input image
5
```



- For task 2a the program groups the features of a particular label to create a mean vector for all the labels and calculates the similarity between the extracted feature vector and all mean label vectors calculated.
- For task 2b the program generates features specific to the chosen category and calculates the similarity between the extracted feature vector and all other vectors for the RESNET 50 neural network model.
- It then retrieves the top k minimum similarities.
- Finally, it presents the output in the following format:

```
K similar labels to input image: 0
label: 0 score: 0.7776090036555232
label: 1 score: 0.7343699328046358
label: 64 score: 0.4238932532732441
label: 6 score: 0.41940500736119396
label: 99 score: 0.3964110225768662
label: 45 score: 0.39190018680892263
label: 63 score: 0.39011574196880205
label: 85 score: 0.37843222032241675
label: 96 score: 0.3691022120637004
label: 65 score: 0.36654833719162333
```

```
K similar labels to input image: 0
label: 0 score: 0.631568988005631
label: 1 score: 0.47996058984040796
label: 99 score: 0.21316804499490813
label: 64 score: 0.19890153794663729
label: 43 score: 0.170070625527428
label: 65 score: 0.1633964674715031
label: 83 score: 0.15318018029968325
label: 6 score: 0.14608425873033032
label: 67 score: 0.1381146107196673
label: 48 score: 0.13233877478211548
```

- **Task 3 (LS1): Top-k Latent Semantics Extraction and Storage**

    - Navigate to the directory containing the Python scripts for Task 3 (LS1), which is present according to the directory structure provided previously.
    - Execute the program using a Python interpreter. For example:

```
python task3.py
```

```
Please pick one of the below options
1. HOG
2. Color Moments
3. Resnet Layer 3
4. Resnet Avgpool
5. Resnet FC
6. Resnet
1
Select K to find K similar images to given input image
5
Select any of the following dimensionality reduction technique
1. SVD
2. NNMF
3. LDA
4. k-means
1
```

    - The program will prompt you for specific inputs, including the choice of feature model(HOG, CM, L3, AvgPool, FC, RESNET), value of k, and dimensionality reduction technique (SVD, NNMF, LDA, or K-Means).
    - The program will process the data, generate latent semantics, and save the results in the specified output file, according to the input provided.

      Example:

      Suppose we are executing Task 3 (LS1) with the following input:

        - Feature Model: HOG
        - Value of k: 5
        - Dimensionality Reduction Technique: LDA

          The program will create a file named `LDA_5.pkl` in the directory `Outputs/T3/HOG/`.

          The generated latent semantics will be stored in `LDA_5.pkl`.

- The latent semantics will be printed on the terminal, providing insight into the label relationships.

```
image_id: 2074
latent feature: 4  latent feature value: 1.8090840795816268
latent feature: 1  latent feature value: 1.299884173734949
latent feature: 2  latent feature value: 0.3384306898486085
latent feature: 0  latent feature value: 0.2649451508053826
latent feature: 3  latent feature value: -0.7132466722638625
image_id: 2076
latent feature: 1  latent feature value: 3.5203711322362583
latent feature: 2  latent feature value: 2.3847177373212167
latent feature: 4  latent feature value: 1.9324845455769295
latent feature: 0  latent feature value: 1.3336962380425552
latent feature: 3  latent feature value: -3.6837651913513407
```

- **Task 4 (LS2): Top-k Latent Semantics Extraction using CP-decomposition**

To run the program to execute the CP-decomposition, the user needs to have tensorly installed.

"pip3 install tensorflow"

The user also needs to have

- Navigate to the directory containing the Python scripts for Task 4 (LS4).
- Execute the program using python. For example:

```
python task4.py
```

```
Input a feature model from the following list:
 CM, HOG, AvgPool, L3, FC, RESNET
Input a value k
```

- The program will prompt the user for two inputs: a feature model and value k.
- After the user enters the inputs, the program will save the output to a CSV named t4_{feature model}_{k}.csv.

Example:

Suppose we are executing Task 4 (LS2) with the following input:

- Feature Model: HOG

- Value of k: 5

The program will create a file named t4_HOG_5.csv

Output pictures

- **Task 5 (LS3) Generating Latent Semantics from Label Similarities**
  - Navigate to the directory containing the Python scripts for Task 5 (LS3), which is present according to the directory structure provided previously.
  - Execute the program using a Python interpreter. For example:

```
python task5.py
```

  - The program will prompt you for specific inputs, including the choice of feature model(HOG, CM, L3, AvgPool, FC, RESNET), value of k, and dimensionality reduction technique (SVD, NNMF, LDA, or K-Means).

```
Enter the feature model
1. HOG
2. ColorMoments
3. ResNet_Layer3_1024
4. ResNet_AvgPool_1024
5. ResNet_FC_1000
Feature model selected: 1
Enter the value of k: 5
Choose dimensionality reduction technique
1. SVD
2. NMF
3. LDA
4. K-Means
Choose dimensionality reduction selected: 1
Calculate a label-label similarity matrix again?
[y|Y/n|N]:n
```

  - The program will process the data, generate latent semantics, and save the results in the specified output file, according to the input provided.

    Example:

    Suppose we are executing Task 5 (LS3) with the following input:

    - Feature Model: HOG
    - Value of k: 5

- Dimensionality Reduction Technique: LDA

  The program will create a file named `LDA_5.pkl` in the directory `Outputs/T5/HOG/`.

  The generated latent semantics will be stored in `LDA_5.pkl`.

- The latent semantics will be printed on the terminal, providing insight into the label relationships.

```
 1.01285011 0.89312792 0.98415279 0.93223761]
Dominant Features: [1.01373788 1.01285011 0.98415279 0.94907817 0.94363332 0.9430471
 0.93223761 0.89312792 0.8626984  0.85093084]
Dominant Feature Number: [0 6 8 2 4 5 9 7 1 3]
----------------------------
Original Features: [1.39766241 1.19827124 1.2977321  1.16985091 1.22989592 1.40163145
 1.32700529 1.22486843 1.60768157 1.48958311]
Dominant Features: [1.60768157 1.48958311 1.40163145 1.39766241 1.32700529 1.2977321
 1.22989592 1.22486843 1.19827124 1.16985091]
Dominant Feature Number: [8 9 5 0 6 2 4 7 1 3]
----------------------------
Original Features: [1.29598895 1.12016294 1.27864262 1.10565203 1.18472607 1.40028773
 1.24705704 1.26800866 1.58521967 1.46468341]
Dominant Features: [1.58521967 1.46468341 1.40028773 1.29598895 1.27864262 1.26800866
 1.24705704 1.18472607 1.12016294 1.10565203]
Dominant Feature Number: [8 9 5 0 2 7 6 4 1 3]
```

- **Task 6 (LS4) Generating Latent Semantics from Image Similarities**
  - Navigate to the directory containing the Python scripts for Task 6 (LS4), which is present according to the directory structure provided previously.
  - Execute the program using a Python interpreter. For example:

```
python task6.py
```

```
Enter the feature model
1. HOG
2. ColorMoments
3. ResNet_Layer3_1024
4. ResNet_AvgPool_1024
5. ResNet_FC_1000
Feature model selected: 1
Enter the value of k: 5
Choose dimensionality reduction technique
1. SVD
2. NNMF
3. LDA
4. kmeans
Choose dimensionality reduction selected: 1
Calculate a image-image similarity matrix again?
[y|Y/n|N]:n
```

- The program will prompt you for specific inputs, including the choice of feature model(HOG, CM, L3, AvgPool, FC, RESNET), value of k, and dimensionality reduction technique (SVD, NNMF, LDA, or K-Means).
- The program will process the data, generate latent semantics, and save the results in the specified output file, according to the input provided.

  Example:

  Suppose we are executing Task 6 (LS4) with the following input:

  - Feature Model: AvgPool
  - Value of k: 5
  - Dimensionality Reduction Technique: SVD

  The program will create a file named `SVD_6.pkl` in the directory `Outputs/T6/AvgPool/`.

  The generated latent semantics will be stored in `SVD_6.pkl`.

- The latent semantics will be printed on the terminal, providing insight into the label relationships.

```
Dominant Features: [38.2013868  36.74364167 35.70636841 35.58576547 35.51597848 35.06908658
 34.94812283 34.06610817 32.3198525  31.99274674]
Dominant Feature Number: [9 0 8 7 2 4 1 3 5 6]
----------------------------
Original Features: [39.46525479 37.78450018 36.31570704 35.68518648 36.19276186 34.44016343
 33.80853114 38.04357887 36.67443717 40.43187695]
Dominant Features: [40.43187695 39.46525479 38.04357887 37.78450018 36.67443717 36.31570704
 36.19276186 35.68518648 34.44016343 33.80853114]
Dominant Feature Number: [9 0 7 1 8 2 4 3 5 6]
----------------------------
Original Features: [38.92015915 37.35647508 37.71785961 36.64610275 36.86981857 34.33576264
 33.8801883  37.722017   36.76285949 40.98017972]
Dominant Features: [40.98017972 38.92015915 37.722017   37.71785961 37.35647508 36.86981857
 36.76285949 36.64610275 34.33576264 33.8801883 ]
Dominant Feature Number: [9 0 7 2 1 4 8 3 5 6]
```

- **Task 7: Image Retrieval Using Image Latent Semantic Representations**

    - Navigate to the directory containing the Python scripts for Task 7, which is present according to the directory structure provided previously.
    - Execute the program using a Python interpreter. For example:

```
python task7.py
```

```
Enter image ID between 0-8676 or image path.
0
Please enter latent semantic in format Task-FeatureModel-ReducedDimension-DimensionReductionTechnique
Valid Tasks: T3, T4, T5, T6
Valid Feature Model: CM, HOG, AvgPool, L3, FC, RESNET
Valid Reduced Dimension: 1 - length of feature model
Valid Dimension Reduction Technique: SVD, NNMF, LDA, kmeans
T3-FC-5-SVD
Select K to find K similar images to given input image
5
```

    - The program will prompt you for specific inputs, including the choice of feature model(HOG, CM, L3, AvgPool, FC, RESNET), value of k, and dimensionality reduction technique (SVD, NNMF, LDA, or K-Means) and task number.
    - The program will process the data (pickle file generated in task 4, 5, 6 & 7), and retrieve top k images and visualize them

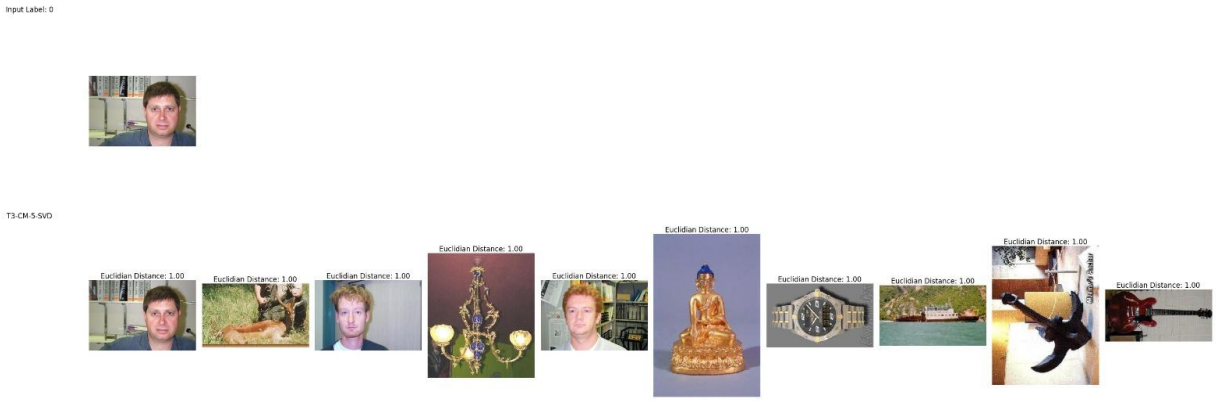
        Example:

        Suppose we are executing Task 7 with the following input:

- Task: example – 3
- Feature Model: example – FC
- Value of k: example – 5
- Dimensionality Reduction Technique: example – SVD

For sample input provided by professor

:



- **Task 8: Image Matching Using Label Latent Semantic Representations**

  - Navigate to the directory containing the Python scripts for Task 8, which is present according to the directory structure provided previously.
  - Execute the program using a Python interpreter. For example:

```
python task8.py
```

```
Enter image ID between 0-8676 or image path.
0
Please enter latent semantic in format Task-FeatureModel-ReducedDimension-DimensionReductionTechnique
Valid Tasks: T3, T4, T5, T6
Valid Feature Model: CM, HOG, AvgPool, L3, FC, RESNET
Valid Reduced Dimension: 1 - length of feature model
Valid Dimension Reduction Technique: SVD, NNMF, LDA, kmeans
T4-CM-5-SVD
Select K to find K similar images to given input image
5
```

  - The program will prompt you for specific inputs, including the choice of feature model(HOG, CM, L3, AvgPool, FC, RESNET), value of k, and dimensionality reduction technique (SVD, NNMF, LDA, or K-Means) and task number.

- The program will process the data (pickle file generated in task 4, 5, 6 & 7), and retrieve top k images and visualize them

  Example:

  Suppose we are executing Task 8 with the following input:

  - Task: 3
  - Feature Model: FC
  - Value of k: 5
  - Dimensionality Reduction Technique: SVD

For the sample input provided by professor we will get below output:

```
K similar labels to input image:


label: 1 score: 0.9390731912943621

label: 0 score: 0.9389979409475668

label: 3 score: 0.9384679287202239

label: 47 score: 0.9384507392977011

label: 43 score: 0.9384495736118111

label: 83 score: 0.9384309472536931

label: 33 score: 0.9384276314525626

label: 95 score: 0.9384222376373041

label: 89 score: 0.9384100684012622

label: 60 score: 0.9383961027291919
```

● Task 9: Label Matching Using Label Latent Semantics

- Navigate to the directory containing the Python scripts for Task 9, which is present according to the directory structure provided previously.
- Execute the program using a Python interpreter. For example:

```
python task9.py
```

```
Enter the feature model
1. HOG
2. ColorMoments
3. ResNet_Layer3_1024
4. ResNet_AvgPool_1024
5. ResNet_FC_1000
Feature model selected: 5
Enter the value of k: 10
Choose dimensionality reduction technique
1. SVD
2. NNMF
3. LDA
4. kmeans
Choose dimensionality reduction selected: 1
Please enter the label(0-100): 0
Please enter the task number(3, 4, 5, 6): 3
Please enter the reduced dimension of latent space: 5
```

- The program will prompt you for specific inputs, including the choice of feature model(HOG, CM, L3, AvgPool, FC, RESNET), value of k, and dimensionality reduction technique (SVD, NNMF, LDA, or K-Means) and task number.
- The program will process the data (pickle file generated in task 4, 5, 6 & 7), and retrieve top k images and visualize them

Example:

Suppose we are executing Task 9 with the following input:

- Task: 3
- Feature Model: FC
- Value of k: 5
- Dimensionality Reduction Technique: SVD

For the above input we will get below output:

```
Top 5 matching labels for label 0:
Label 0 (Similarity Score: 0.0)
Label 58 (Similarity Score: 2.32)
Label 98 (Similarity Score: 12.06)
Label 49 (Similarity Score: 12.43)
Label 1 (Similarity Score: 16.35)
```

Note: Here we have used euclidean distance

- ## Task 10: Image Matching Using Label Latent Semantics

    - Navigate to the directory containing the Python scripts for Task 10, which is present according to the directory structure provided previously.
    - Execute the program using a Python interpreter. For example:

```
python task10.py
```

```
1. ColorMoments
2. HOG
3. ResNet_AvgPool_1024
4. ResNet_Layer3_1024
5. ResNet_FC_1000
Feature model selected: 5
Enter the value of k: 8
Choose dimensionality reduction technique
1. SVD
2. NNMF
3. LDA
4. kmeans
Choose dimensionality reduction selected: 1
Please enter the label(0-100): 9
Please enter the task number(3, 4, 5, 6): 3
Please enter the reduced dimension of latent space: 5
```

    - The program will prompt you for specific inputs, including the choice of feature model(HOG, CM, L3, AvgPool, FC, RESNET), value of k, and dimensionality reduction technique (SVD, NNMF, LDA, or K-Means) and task number.
    - The program will process the data (pickle file generated in task 4, 5, 6 & 7), and retrieve top k images and visualize them

        Example:

        Suppose we are executing Task 10 with the following input:

        - Task: 3
        - Feature Model: FC
        - Value of k: 5
        - Dimensionality Reduction Technique: SVD

For the above input we will get below output:





- **Task 11: Page rank & personalized page rank**

    - Navigate to the directory containing the Python scripts for Task 11
    - Execute the program using a Python interpreter. For example:

      `python task11.py`

    - The program will prompt you for specific inputs as mentioned in the execution instructions. Here is a screenshot of the prompt:

```
Files already downloaded and verified
Enter the feature model
1. ColorMoments
2. HOG
3. ResNet_AvgPool_1024
4. ResNet_Layer3_1024
5. FC
Feature model selected: 1
Enter the value of m: 5
Enter the value of n: 10
Please enter the label(0-100): 0
Press 1 if you want to use a feature model, press 2 if you want to use a latent space2
Enter the value of k: 5
Choose dimensionality reduction technique
1. SVD
2. NNMF
3. LDA
4. kmeans
Choose dimensionality reduction selected: 1
Please enter the task number(3, 4, 5, 6): 3
Does a new similarity matrix need to be computed, type y for yes: n
```

    - The program will process the data, generate and print the similarity graph, and print the ‘m’ most similar images for a specific label. A screenshot for each of the n nearest nodes for a few image ids is attached for each of the following examples.

      Example:

      Suppose we are executing Task 11 with the following input:

      - Task: 3

- Feature Model: Color Moments
- Value of k: 5
- Dimensionality Reduction Technique: SVD
- Value of m: 5
- Value of n: 10
- Label: 0

Example:

Suppose we are executing Task 11 with the following input:

- Task: 3
- Feature Model: Color Moments
- Value of k: 5
- Dimensionality Reduction Technique: SVD
- Value of m: 5
- Value of n: 10
- Label: 20

```
Node pairs for image 0
(0,2576)
(0,124)
(0,1950)
(0,17)
(0,1647)
(0,4139)
(0,2475)
(0,1372)
(0,2329)
(0,4137)
Node pairs for image 1
(1,1497)
(1,2454)
(1,2263)
(1,1103)
(1,3913)
(1,3578)
(1,989)
(1,3763)
(1,2601)
(1,2961)
Node pairs for image 2
(2,64)
(2,4095)
(2,2925)
(2,3088)
(2,302)
(2,370)
(2,2512)
(2,281)
(2,4094)
(2,233)
Node pairs for image 3
(3,2981)
(3,3385)
(3,2312)
(3,3085)
(3,2290)
(3,3027)
(3,2397)
(3,4222)
```

Example:

Suppose we are executing Task 11 with the following input:

- Task: 3
- Feature Model: Color Moments
- Value of k: 5
- Dimensionality Reduction Technique: SVD
- Value of m: 5
- Value of n: 10
- Label: 55

```
Node pairs for image 0
(0,2576)
(0,124)
(0,1950)
(0,17)
(0,1647)
(0,4139)
(0,2475)
(0,1372)
(0,2329)
(0,4137)
Node pairs for image 1
(1,1497)
(1,2454)
(1,2263)
(1,1103)
(1,3913)
(1,3578)
(1,989)
(1,3763)
(1,2601)
(1,2961)
Node pairs for image 2
(2,64)
(2,4095)
(2,2925)
(2,3088)
(2,302)
(2,370)
(2,2512)
(2,281)
(2,4094)
(2,233)
Node pairs for image 3
(3,2981)
(3,3385)
(3,2312)
```

Example:

Suppose we are executing Task 11 with the following input:

- Task: 3
- Feature Model: Color Moments
- Value of k: 5
- Dimensionality Reduction Technique: SVD
- Value of m: 5
- Value of n: 10
- Label: 100

```
Node pairs for image 0
(0,2576)
(0,124)
(0,1950)
(0,17)
(0,1647)
(0,4139)
(0,2475)
(0,1372)
(0,2329)
(0,4137)
Node pairs for image 1
(1,1497)
(1,2454)
(1,2263)
(1,1103)
(1,3913)
(1,3578)
(1,989)
(1,3763)
(1,2601)
(1,2961)
Node pairs for image 2
(2,64)
(2,4095)
(2,2925)
(2,3088)
(2,302)
(2,370)
(2,2512)
(2,281)
(2,4094)
(2,233)
Node pairs for image 3
(3,2981)
(3,3385)
(3,2312)
```

Example:

Suppose we are executing Task 11 with the following input:

- Feature Model: Color Moments
- Value of m: 5
- Value of n: 10
- Label: 0

```
Node pairs for image 0
(0,78)
(0,110)
(0,202)
(0,9)
(0,242)
(0,18)
(0,11)
(0,191)
(0,2)
(0,293)
Node pairs for image 1
(1,10)
(1,149)
(1,365)
(1,160)
(1,361)
(1,367)
(1,156)
(1,225)
(1,150)
(1,161)
Node pairs for image 2
(2,9)
(2,13)
(2,183)
(2,203)
(2,233)
(2,237)
(2,242)
(2,70)
(2,211)
(2,231)
Node pairs for image 3
(3,401)
(3,9)
```

Example:

Suppose we are executing Task 11 with the following input:

- Feature Model: Color Moments
- Value of m: 5
- Value of n: 10
- Label: 20

```
Node pairs for image 0
(0,78)
(0,110)
(0,202)
(0,9)
(0,242)
(0,18)
(0,11)
(0,191)
(0,2)
(0,293)
Node pairs for image 1
(1,10)
(1,149)
(1,365)
(1,160)
(1,361)
(1,367)
(1,156)
(1,225)
(1,150)
(1,161)
Node pairs for image 2
(2,9)
(2,13)
(2,183)
(2,203)
(2,233)
(2,237)
(2,242)
(2,70)
(2,211)
(2,231)
Node pairs for image 3
(3,401)
```

Example:

Suppose we are executing Task 11 with the following input:

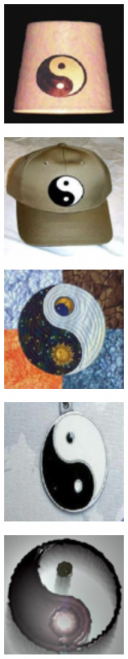
- Feature Model: Color Moments
- Value of m: 5
- Value of n: 10
- Label: 55

```
Node pairs for image 0
(0,78)
(0,110)
(0,202)
(0,9)
(0,242)
(0,18)
(0,11)
(0,191)
(0,2)
(0,293)
Node pairs for image 1
(1,10)
(1,149)
(1,365)
(1,160)
(1,361)
(1,367)
(1,156)
(1,225)
(1,150)
(1,161)
Node pairs for image 2
(2,9)
(2,13)
(2,183)
(2,203)
(2,233)
(2,237)
(2,242)
(2,70)
(2,211)
(2,231)
Node pairs for image 3
(3,401)
(3,9)
```

Example:

Suppose we are executing Task 11 with the following input:

- Feature Model: Color Moments
- Value of m: 5
- Value of n: 10
- Label: 100



```
Node pairs for image 0
(0,78)
(0,110)
(0,202)
(0,9)
(0,242)
(0,18)
(0,11)
(0,191)
(0,2)
(0,293)
Node pairs for image 1
(1,10)
(1,149)
(1,365)
(1,160)
(1,361)
(1,367)
(1,156)
(1,225)
(1,150)
(1,161)
Node pairs for image 2
(2,9)
(2,13)
(2,183)
(2,203)
(2,233)
(2,237)
(2,242)
(2,70)
(2,211)
(2,231)
Node pairs for image 3
(3,401)
(3,9)
```

Conclusions

- **Analysis of Task 0**

  Task 0.a employs a pre-trained RESNET50 model to extract diverse features from even-numbered images in the Caltech101 dataset. This includes Color Moments, Histograms of Oriented Gradients, and various ResNet-based representations. By focusing on even-numbered images, the task hones in on specific data subsets. Crucially, it stores both image IDs and their corresponding feature vectors, alongside original labels, enabling context preservation.

  Task 0.b entails the development of a program that takes user inputs: an image ID or file, a selected feature space, and a positive integer 'k'. The program then generates features based on the chosen category and calculates similarity scores between the extracted feature vector and all other vectors stored in the database. The objective is to identify the 'k' most similar images.

- **Analysis of Task 1**

  By leveraging the chosen feature space and user-provided parameters, the program successfully narrows down the search and provides a curated selection of images that are most relevant to the queried label, using feature mean vector. This functionality is invaluable for tasks like content-based image retrieval, where users need to find images associated with particular categories or labels. The visualization of the top 'k' images, accompanied by their respective scores, offers a clear and intuitive representation of the program's performance.

- **Analysis of Task 2**

  The results of Task 2.a showcase the program's ability to efficiently match query images to likely corresponding labels based on user-defined criteria. By inputting a query image ID or file, along with a chosen feature space and positive integer 'k', the program successfully identifies and lists 'k' most probable matching labels. This functionality provides a powerful tool for categorizing and organizing images within the dataset.

  Upon analysis, it was observed that the program successfully generated accurate label predictions for the query images. The model demonstrated a high level of proficiency in recognizing and associating images with relevant labels, as the similarity is maximum in case of the category that the image actually belongs to.

- **Analysis of Task 4**

  By employing CP-decomposition, the program decomposes the tensor into its constituent parts, revealing hidden patterns and relationships between images, features,

and labels. The resulting latent semantics are presented in the form of lists of label-weight pairs, ordered by weight, providing a clear understanding of the significance of each latent semantic.

- **Analysis of Task 3, 5, 6**

Depending on the type of input matrix, the resulting output will vary. Specifically, for image-image similarity, the matrix will be of size 4339x4339. In the case of label-label similarity, it will be of size 101x101. When dealing with image-feature similarity, the input matrix will be of size 4339xm, with 'm' representing the number of features for the chosen feature descriptor.

Singular Value Decomposition (SVD):

Output Dimensions: The resulting latent semantics from SVD will have dimensions corresponding to the user-defined value 'k'.

Data Representation: The latent semantics extracted using SVD represent the most significant patterns or features in the image similarity matrix. These patterns are linear combinations of the original images.

Interpretation: Each latent semantic is a weighted combination of images, and higher weights indicate a stronger influence of those images on the semantic. These semantics can be interpreted as identifying underlying themes or commonalities among the images.

Non-negative Matrix Factorization (NNMF):

Output Dimensions: The latent semantics obtained through NNMF will have dimensions corresponding to the user-defined value 'k'.

Data Representation: NNMF enforces non-negativity in both the input and output matrices, which can be particularly useful for datasets where negative values don't have a meaningful interpretation.

Interpretation: The latent semantics derived from NNMF capture positive patterns or features in the image similarity matrix. These patterns represent additive contributions from different images, allowing for a more intuitive interpretation.

Latent Dirichlet Allocation (LDA):

Output Dimensions: The resulting latent semantics from LDA will have dimensions equal to the number of specified topics (user-defined 'k').

Data Representation: LDA is a probabilistic model that assigns topics to images based on their content. The latent semantics in this context represent the distribution of topics across images.

Interpretation: Each latent semantic corresponds to a topic, and the weights indicate the likelihood of an image belonging to that topic. This can be valuable for uncovering thematic clusters or content categories within the dataset.

K-means Clustering:

Output Dimensions: The latent semantics from K-means clustering will have dimensions corresponding to the number of clusters specified by the user ('k').

Data Representation: K-means clustering groups images into clusters based on their similarity. The latent semantics represent the centroid of each cluster.

Interpretation: Each latent semantic is a centroid vector that characterizes a cluster of images. It provides a compact representation of the cluster's characteristics, allowing for efficient analysis and comparison.

- **Analysis of Task 7, 8, 9, 10**

Below is the analysis of the results for each dimensionality reduction technique:

- SVD:
  - The images retrieved under the SVD-based latent space exhibit a balance between capturing the most significant features and minimizing information loss. This technique emphasizes the dominant patterns in the data. Therefore, the images returned in this scenario are likely to highlight the most pronounced characteristics shared with the input image. Most dominant feature are related to lower index in the diagonal matrix of $\sum$.
- NNMF:
  - NNMF focuses on non-negative data and can be particularly effective in cases where the underlying features are inherently positive. The images retrieved using NNMF-based latent semantics are likely to emphasize positive correlations and patterns within the data. This can lead to the identification of images that share prominent positive attributes with the input image. The more we keep on looping the iterations to reduce loss the better we are able to retrieve the results.
- LDA:
  - LDA aims to maximize the separation between classes in the reduced feature space. As a result, images retrieved using LDA-based latent semantics are

expected to be distinctive in terms of class separability. This technique may excel in scenarios where the images belong to distinct categories, and the user is interested in finding visually distinguishable matches.

- k-means:
    - The k-means technique partitions the feature space into clusters and identifies centroids. Images retrieved using k-means-based latent semantics are likely to be representative of the clusters to which the input image belongs. This approach may be particularly useful when the user is interested in finding images that share similar cluster characteristics. This worked best, at it used clustering methods, and looping enough times will lead to correct clustering which is equivalent to categorizing it in the labels provided.

Import Analysis: In this project, we aimed to reduce large feature vectors to much smaller dimensions, which inevitably leads to some information loss. For instance, when we compressed a 1000-dimensional vector into just 5 dimensions, there was a significant reduction in the richness of information. This reduction could lead to inconsistencies when compared to the original input image.

However, an intriguing observation is that despite the loss of detail, the output consistently belonged to the same category. For example, when using SVD decomposition in Task 10 with the query label "face", we consistently received results featuring cell phone. This indicates that the reduced dimensions do capture some critical characteristics of the original images, particularly those essential for distinguishing between different classes or categories.

While the reduced dimensions correctly retain class-related information, the issue arises when trying to map a new feature with a large vector size to a smaller dimension. This process inevitably leads to some loss of information, as the finer details get compressed. This phenomenon highlights the trade-off between dimensionality reduction and information retention, which is a crucial consideration in tasks involving feature extraction and reduction.

Furthermore, during our experimentation, we observed that adjusting parameters can have a significant impact on the quality of the output. For example, when we increased the number of iterations and latent features to 50 for the NNMF (Non-negative Matrix Factorization) approach in Task 9, the results became remarkably similar to those obtained from label-label similarity computations.

This trial-and-error process in parameter tuning showcases the sensitivity of the chosen methods to these settings. It's evident that finding the right combination of parameters is crucial for obtaining accurate and meaningful results. This process involves determining a "soft spot" in the parameter space where the model performs optimally for a given task. This iterative approach to parameter selection is a common practice in machine learning and data analysis.

- **Analysis of Task 11**

  Overall, in terms of finding the n-most relevant images, the personalized page rank performed much similar images on the latent space T3-CM-5-SVD than on a feature space. There could be several reasons for this, it is likely that using a dimensionality reduction technique such as SVD removed redundancy from the feature set, reducing noise. This is evident from the fact that all the m relevant images for a label had similar colors. In terms of the similarity graph, the n most similar images for feature space were more closely related than on a latent space. This could be because when creating a similarity matrix, having more features provided more information. However, since the outputs that task 11 was tested on contained two different features spaces (Color Moments and ResNet FC), it cannot be said with certainty that performing on latent space or feature space was better. Overall it performed well in both spaces.

Related Work

One of the chosen dimensionality reduction techniques implemented was LDA. According to prior research, linear discriminant analysis is a better method for facial recognition, especially in comparison to the method PCA. LDA was shown to have high recognition rates and outperformed even in a lower dimensional space (Belhumeur et al. 1997).While we do not use PCA in our project, we use SVD which is closely related to PCA as SVD is often used to help implement PCA. The results in our work also aim to compare LDA with other dimensionality techniques.

Our work also implements CP-decomposition to find latent semantics. More specifically, one of our sample inputs requires using the “HOG” feature model in combination with CP-decomposition to find latent semantics to use in identifying images. This idea has previously been tested and results show promise in image classification (Vo et al. 2013). The performance indicates that tensor decomposition can be used in a way to improve the effectiveness of HOG image descriptors. We attempt to replicate these results with our work in the phase.

References

B. Chaudhuri, B. Demir, S. Chaudhuri and L. Bruzzone, "Multilabel Remote Sensing Image Retrieval Using a Semisupervised Graph-Theoretic Method," in *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 2, pp. 1144-1158, Feb. 2018, doi: 10.1109/TGRS.2017.2760909.

Belhumeur, P. N., Hespanha, J. P., & Kriegman, D. J. (1997). Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(7), 711-720.

Huang, S., Li, X., Candan, K. S., Sapino, M. L. (2016). Reducing seed noise in personalized PageRank. Social Network Analysis and Mining, 6(1), 1-25

Li, W., Liu, A., Nie, W., Song, D., Li, Y., Wang, W., Xiang, S., Zhou, H., Bui, N.-M., Cen, Y., Chen, Z., Chung-Nguyen, H.-H., Diep, G.-H., Do, T.-L., Doubrovski, E. L., Duong, A.-D., Geraedts, J. M. P., Guo, H., Hoang, T.-H., … Zhao, S. (1970, January 1). *Monocular image based 3D model retrieval*. Eurographics DL Home. https://diglib.eg.org/handle/10.2312/3dor20191068

Suresh, S., Mohan, S. ROI-based feature learning for efficient true positive prediction using convolutional neural network for lung cancer diagnosis. *Neural Comput & Applic* 32, 15989–16009 (2020). https://doi.org/10.1007/s00521-020-04787-w

Vo, Tan & Tran, Dat & Ma, Wanli & Nguyen, Khoa. (2013). Improved HOG Descriptors in Image Classification with CP Decomposition. 8228. 384-391. 10.1007/978-3-642-42051-1_48.

Appendix

- **Task 0** is a common task that needs to be performed by every subgroup. Every member of the project team is responsible for executing Task 0. It involves mapping even-numbered labeled images from the Caltec101 dataset into five different feature spaces and storing the resulting data vectors.
- **Task 1 & 2** are independent tasks, meaning they can be worked on separately. This implies that the team can divide themselves to handle these tasks in parallel. Each task involves different operations: Task 1 is about querying relevant images based on a user-specified label and feature space, while Task 2 focuses on matching labels to a query image.
- **Task 3 & 4** were specifically designated to Ariana, Rajat, and Kasi. These three individuals are responsible for completing Tasks 3 and 4. Task 3 entails extracting top-k latent semantics using various dimensionality reduction techniques, while Task 4 involves obtaining latent semantics using CP-decomposition of a three-modal tensor.
- **Task 5 & 6** Yash, Maryam, and Peter are responsible for the completion of Tasks 5 and 6. Task 5 focuses on generating a label-label similarity matrix and storing it in the database. Task 6 involves visualizing the similarity matrix through clustering and plotting.
- **Task 7 & 8** were designated to Peter, Maryam, and Yash. Task 7 involves implementing a program that identifies and visualizes similar images based on selected latent semantics. Task 8 involves listing likely matching labels based on user-provided parameters.
- **Task 9 & 10** were assigned to Ariana, Rajat, and Kasi. Task 9 involves label matching using latent semantics, while Task 10 focuses on label matching using a neural network model.

The group worked on the tasks in a timely manner and we set small deadlines for each task. Using the time allotted, the group worked together to complete their designated tasks and were able to complete the phase as a whole. The group also worked on the report together and wrote about each task and described the importance of each task and described the tasks.