CSE 515 Phase 1 Report

Group Members from Group 11:
Tanner Greenhagen,
Connor Wardell,
Varad Joshi,
Mounish Chennupati,
Khushank Goyal

Abstract

High dimensions of data is a challenge to media in a dynamic world. It increases the storage, computation and cost of indexing. Reducing the dimensions of data is vital for effective and efficient search and analysis. Graphs are more rich and expressive forms of data representation. They represent a pairwise relationship between the features of a given object. In this phase we will experiment with dimensionality techniques like PCA, SVD and report the top k latent semantics. Additionally, we create similarity graphs based on the relational properties and return similar subjects using ASCOS++ measure.

Keywords: Extraction, Matching, Similarity, Feature, Color Moment, Extended Local Binary Patterns, Histogram of Oriented Gradients, dimensional reduction, principal component analysis, singular value decomposition, graphs, ASCOSS++

1 Introduction

This work centered on taking image data and reducing the demensionality of the images into a smaller amount so that common distance measures could be used to find similar images, the type of the image, or the subject in the image. The original features of an image came from calculating the color moment, the extended local binary pattern (ELBP), or the histogram of oriented gradients (HOG). The color moment of an image is used to distinguish one image from another based on the color orientation of that image [5]. It is a commonly used feature for image retrieval and image similarity comparison. For the purposes of this work, it was composed of the mean, standard deviation, and third root of the skewness of each 8x8 pixel block of each image. There can then by weights applied to each of the three quantities to affect their impact, but in this case they all were given equal weight. The Extended Local Binary Patterns is a less common measurement that takes the local binary pattern and divides it by the variance surrounding the pixel. For the regular local binary patterns, each pixel is examined individually and checked against its neighboring pixels to determine if the value of its neighbor's color value is greater or less than [12]. A specified number of locations are checked (8 in the simplest case) a certain number of spaces from the pixel (the radius from the checked pixel to the compared pixels). The neighboring pixels are checked either in a clockwise or counterclockwise fashion and for each pixel with a greater gray scale value a 1 is used and a 0 is used for the pixels of lesser gray scale value. This generates a binary number than is then converted into decimal as the value for the pixel being surrounded. The extended part of this comes from using rotational invariant local binary patterns with a uniform method and dividing the value by the variance of the surrounding neighbors [10]. The uniformity refers to the spatial transitions of the binary number causing cases where there were two or less spatial transitions differently than if there were more than two. The case where there were more than two spatial transitions had the value set to be one greater than the number of neighbors checked which was not the normal case for the default local binary patterns. The local gray scale variance is then used to create a joint distribution with with the uniform local binary pattern (ULBP/VAR). The histograms of oriented gradients are based off of the distribution and the value of the gradients of an image [2]. Similar images should have similar changes in pixels that display a kind of figure. The gradients are calculated and then stored by what bin they fall under of the direction they most closely pointed to.

Those feature models are the same as they were in the last phase, but in addition we now look at the dimensionality reduction techniques. Dimensional reduction in our case is done by finding the finding latent semantics that create a new set of basis vectors that are less in number than the original dimensions of the image feature matrix. The first dimensionality reduction technique is Principal Component Analysis or PCA [3]. It is the dimensionality reduction technique where the covariance matrix is factorized. Covariance matrix (C) is square symmetric and undergoes eigen decomposition. Post decomposition it results in $C = U * S * V^T$ **where U is the left factor matrix. V is the right factor matrix and in this case, since the covariance matrix is square with its dimensions being the the number of features by the number of features, V is just the transpose of U. S is the core diagonal matrix which consists of the values of square roots of eigenvalues(discrimination power) in descending order. Setting the smallest eigen values to 0 will then reduce the number of dimensions and remove them in order of the variance they provide. We want to keep the latent semantics that provide the most variance as they are the best at discriminating between different object (images). The other dimensionality reduction technique we implemented was Singular Value Decomposition or SVD [7]. The data matrix (D) undergoes factorization into a matrix U which is the eigen vectors of** $D * D^T$ **and a matrix V which comes from the eigen vectors** $D^T * D$. **After decomposition it results in** $D = U * S * V^T$ **where U (object-object matrix) is the left eigenvector of the DDT matrix. V(feature-feature) is the right factored matrix of the eigenvectors of the** $D^T * Dmatrix$. **S is the core diagonal matrix which consists of the values of square roots of eigenvalues which can be order from largest to smallest. The smallest eigen values can then be set to 0 from the**

smallest to the largest to make sure to limit the error to be as small as possible while reducing the dimensions to k.

Another difference from the last project is that instead of just dealing with images of different subjects and facial positions, there is now different image transformations applied to the images which are labeled as different image types. We then create similarity matrices between the types themselves which represent how similar one type is to another type. We also create similarity matrices between different subjects to see which subjects are the most similar to each other. The intersection of these groups (types or subjects) has a score that represents how close the groups are to each other. A lower score would mean more similar while a higher score would mean less similar. The scores can then be ordered to get a ranking of the similarity among the groups.

The goal of this project is to implement programs which illustrate the knowledge and understanding of dimensionality reduction and latent semantics on a given data set. Furthermore, we also create graphs for analysis and similarity search. Using the inputs given, the deliverable will carry out decompositions of data and provide the latent semantics. Also, it will help in graphical analysis. This is achieved by the following tasks. For Task 1 we have to implement a program which works on the three different image features. It's up to the user to select either of the CM or HOG or ELBP. After selecting the feature, the user has to input the values X and 'k' where X is the type of image and k is the number of latent semantics extracted using PCA or SVD. The latent semantics will be presented in the form of a list of subject-weight pairs in descending order of weights. Task 2 is similar to Task 1, but the user has to input Y value instead of X where Y is the subject ID. The latent semantics will be presented in the form of a list of type-weight pairs in descending order of weights. In Task 3 we create a type-type similarity matrix of the features extracted using the three feature models. It's up to the user to select either of the CM or HOG or ELBP. Then we apply the user selected dimensionality reduction techniques on this type-type similarity matrix and report top k latent semantics. The latent semantics will be presented in the form of a list of type-weight pairs in descending order of weights. Task 4 is similar to Task 3, here we create subject-subject similarity matrix instead of type-type similarity matrix. The latent semantics will be presented in the form of a list of subject-weight pairs in descending order of weights. Task 5 users will input a filename of the query image and a latent semantics file. We have to identify and visualize most similar n images under the given latent semantics. For task 6 the user gives similar input, and we have to associate a type label (X) to the image under selected latent semantics. Task 7 is similar to Task 6, but we have to associate a subject ID (Y) to the image under selected latent semantics. In Task 8 we have to create a similarity graph for a given subject-subject similarity matrix using n most similar subject for some user specified k. Additionally we have to identify the most significant m objects in the collection using ASCOS++ measure [1]

There were a few assumptions we have made for this assignment. The first was that all of the images would be .pngs, would all be gray scale, and would all be 64 by 64. Our system is only designed to work with those kinds of images. We also expect to only work with files that are from the original dataset, transformed versions from the original dataset, or images that we can move locally into the images folder [11]. We have also assumed that there are only 12 different types and that there are 40 different subjects. Additionally, have completed only tasks 1 to 8 and only implemented PCA and SVD because one of our members left our team so task 9 along with LDA and k-means was dropped. We have also assumed that for our implementation of the ASCOS++ algorithm the measurement is 0.9 for c.

2 Description of the proposed solution/implementation

For the design decisions, each proposed solution is explained when it was first used. For example, calculating the color moment, something that is used in almost every task, is explained when discussing the solutions for task 1 rather than being repeatedly explained.

For task 1 we first implemented a means to take an image and calculate the features of an inputted type of images. The color moment was calculated by dividing the image into 8 by 8 chunks for a total of 64 separated 8 by 8 blocks. For each of those blocks, the 64 pixels were examined and the mean, standard deviation, and the third root of the skew were calculated. All of the means were stored in one list, all of the standard deviations in another list, and all of skews were stored in a third list making for 3 vectors of length 64. These three lists were then combined into one vector where the first 64 values were the means, the second 64 values were the standard deviations, and the third 64 values were the skews for a vector of length 192. This represented the 192 features calculated to represent the color moment. The ELBP was was calculated by first taking the image pixels and using the skimage library's LBP pattern [13]. We used the uniform method rather than the default to calculate the rationally invariant extended local binary pattern similarly to how it was done by Mdakane [10]. We used a radius of 4 and a number of points of 32 to fully capture a greater number of the points around each pixel and the change in the darkness. We then simplified the number of bins to be 32 based on the variance of overall pixels. This resulted in a 32 long vector that represent the ELBP features. The HOG was calculated using the skimage library again similar to ELBP [13]. We used 8 by 8 pixel cells and 2 by 2 cells per block as the parameters along with the pixels. In return we got a 1764 long vector that represented 1764 different bins which represented the frequency of the gradient being within the bounds of that bin. The next design decision was how to perform PCA and SVD. Before any reduction was performed, the images corresponding to the type requested had their features generated making a o by f matrix (o was number of images, f was number of features). This was then used as the input for PCA. The first step was generating the covariance matrix. We then took the covariance matrix and calculated the eigen values and eigen vectors using the numpy library [6]. This eigen vector matrix was then transposed to get the original right factor matrix. Then each eigen vector with the next highest eigen value associated with it was added to make a right feature matrix that was then only k by f rather than being f by f. This matrix was what was then used for data projection. For SVD, the original matrix D, was multiplied to create a left matrix created by $D * D^T$ and a right matrix represented by $D^T * D$ with dimensions o by o and f by f respectively. The eigen vectors were then taken and sorted by their values to get the k most significant eigen vectors. The eigen vectors of the right matrix were what would be used for projection. The next task was to the get a vector of f length to represent each subject. To represent all 120 images of a single subject, the features were calculated according to the inputted feature model and then were averaged among all 120 images. We reasoned that the average values of the features would be best at representing the subject as a whole. With this, 40 subject feature vectors (there were 40 subjects) were calculated and used to create a 40 by f matrix. This matrix was then projected into the new feature space represented by k latent semantics generated from PCA or SVD. This projected matrix is then 40 by k and is what is saved as the type-weight pairs. It is saved as:
latent_semantics_(dimensional_reduction_technique)_(feature_model)_(subject_number)_(k).txt. The matrix used for projecting data that will be used in tasks 5, 6, and 7 is stored with the same name except with an r in front.

For task 2, most of the design decisions were exactly the same as task one except, instead of creating condensed vectors for the 40 subjects, condensed vectors are created from the average of the features of the 12 types. The final type-weight pairs are stored as a 12 by k matrix as
latent_semantics_(dimensional_reduction_technique)_(feature_model)_(subject_number)_(k).txt.
Task 3 takes in a directory of images, a feature model, a dimensionality reduction technique, and a 'k' value. It then saves a type-type similarity matrix and returns a list of type weight pairs ordered in decreasing order of weights. We first create a type dictionary to store the images of each corresponding types from a directory of images. We then use the provided feature model to calculate the latent semantics of each image and add it to the corresponding type in the type dictionary. Then we initialize a similarity matrix of size 12 X 12 with zeros (as we have 12 types of images). In this similarity matrix, each row and column represents a

type of image. Each cell in this matrix represents the similarity of the image type of the row to that of the column. The similarity matrix is populated by calculating the average distance of the images of the same subject, with different types. To calculate the similarity between two images we are using the euclidean distance measure as we are already reducing the dimensions of the image by using the feature model. The matrix has its diagonal as 0 as one type would be equal(most similar) to itself. Then on the similarity matrix we perform the given dimensionality reduction by also passing the 'k' value. The dimensionality reduction method returns the top 'k' right feature matrix. Then we project the similarity matrix to the right feature matrix to obtain the new latent semantic space. These latent semantics are stored in a file.
Task 4 takes in a directory of images, a feature model, a dimensionality reduction technique and a 'k' value. It then saves a subject-subject similarity matrix and returns a list of subject weight pairs ordered in decreasing order of weights.We first create a type dictionary to map each type to a number for ease of calculations. Then we create a 3 dimensional matrix to store the features of each image and initialize it to 0s. Here the first dimension represents the subject, second dimension represents the type and the third dimension represents the different images of that subject and type. We then iterate over all the images and then use the given feature model to calculate and store the features of the image. We then initialize a 40 X 40 matrix with zeros(as there are 40 subjects). We then populate the similarity matrix by calculating the average distance of the images of the same subject with different types. We use euclidean distance to calculate the distance between two images as the dimensions are already reduced by using the feature model. The matrix has its diagonal as 0 as one type would be equal(most similar) to itself The similarity matrix is then reduced using the dimensionality reduction algorithm provided, which returns the top 'k' right feature matrix. The similarity matrix is then projected onto the right feature matrix to obtain the new latent semantics. These latent semantics are then stored in a file.

For task 5, the biggest design decision was how to compare the distance image features to each other. In the last project, we used metrics such as earth movers distance, angle, and Manhattan distance that did not seem to work very well when the image was transformed (at least with the testing we did). We did however find that the euclidean distance, did happen to perform quite well when comparing the images. Obviously, using euclidean distance itself is quite common and can also be used with some slight variations like image base euclidean distance, but for our purposes the regular euclidean distance worked the best and most appropriate [9][4]. The provided latent semantic file has the name of the feature model used to create it, so that feature model was used to get the features of the individual images. To be able to compare the features of the images, each image was first projected into the vector space that came from one of the latent semantic files. Once all of the images (database images and provided image) were projected, the distance between the most database images and provided query image were calculated and the similarities were ranked. The n most similar images were then displayed via file name and visually. The results of our experimentation seemed very good, as roughly 80 percent of the 5 most similar images returned were of the same type. For some types like rot and noise01 the type accuracy was even higher while types like cc and con were slightly less accurate. If the image was of a different type, it was more likely to be an image of the same subject as the one used (if the image was from the original database) or almost just as often, a completely different image with a very different transformation. This can be explained by the uniqueness of each transformation making the images of the same subject quite different from the same subject under a completely different transformation.
Task 6 takes in a query image, a latent semantics file and a type label x and then returns a type that is associated with the query image. We first read in the semantic data from the latent semantic file and then parse the name of the file to get the specific feature model that was used on the images. We then get the features of the query image using the same feature model that was used on the latent semantic file. We then gather the features of each type by getting the features of each image of each type and then averaging those values for each feature together. We're then left with a matrix that is the number of types by number of features. Next we plot each type's features into the same space as the latent semantics, plot the query image into the same space as the latent semantics, and then take the euclidean distance between those two values for each type. This produces a list that is the number of types long that shows how similar those types are to the query image. We then output those types ranked from most similar to least similar. In general, when the latent semantics file given was one of multiple subjects the results were fairly accurate at producing the correct type of the query image.
Task 7 takes in a query image, a latent semantics file and a subject label y and then returns a subject that

is associated with the query image. We first read in the semantic data from the latent semantic file and then parse the name of the file to get the specific feature model that was used on the images. We then get the features of the query image using the same feature model that was used on the latent semantic file. We then gather the features of each subject by getting the features of each image of each subject and then averaging those values for each feature together. We're then left with a matrix that is the number of subjects by number of features. Next we plot each subject's features into the same space as the latent semantics, plot the query image into the same space as the latent semantics, and then take the euclidean distance between those two values for each type. This produces a list that is the number of subjects long that shows how similar those subjects are to the query image. We then output the subjects ranked from most similar to least similar. In general, when the latent semantics file given was one of multiple types the results were fairly accurate at producing the correct subject of the query image. However, it is worth noting that when the latent semantic file was generated from a set of subject images the results varied wildly. This is thought to have to do with how different the image feature values would be for different types. Every image of a type was relatively similar while the same subject of different types had pixel values that could be drastically different.
Task 8 takes the subject-subject similarity matrix, the number of similar images 'n', and the number of most significant images 'n' and outputs a graph, ASCOS ++ for the graph and the m most significant subjects. In the first part of task 8 we have to create a graph with n most similar subjects for each subject. We first take the subject-subject similarity matrix which we generated in task 4 and store in into a matrix of size 40*40 as we have 40 subjects. Then we calculate the n most similar images for each subject. In the Subject - Subject similarity matrix, if the similarity score is less then the subjects are more similar and if the score is high then are less similar as we have considered the distance function to calculate the similarity matrix. So we select the n least values from the subject- subject similarity matrix for each subject other than the subject itself. Then we generate a graph with 40 subjects as nodes and 40 *n edges. For the weight of the edge, we have taken the inverse of the similarity score we got from the subject-subject similarity matrix.We did this because as mentioned above,in subject-subject similarity matrix generated in task 4, the lesser value means most similarity. But as we are going to calculate the ASCOS++ feature in part 2 of the task, it considers weights of the edges and if the weight is less than the nodes are less similar and more similar if more edge weight, which is the opposite of what subject-subject similarity matrix meant in task 4. So to consider the subject-subject similarity score for task 4 , we took the inverse of the score as the edge weight.

In the second part of task 8, we used the ASCOS ++ measure to calculate the 'm' most significant subjects of the graph we generated from part 1 of task 8. ASCOS score states that the similarity score from node I to j is dependent on the similarity score between the in neighbors of node I to node j [1] . But the ASCOS doesnt consider the weights of the edges. The ASCOS ++ score tries to improve the ASCOS score by considering the whole topology of the graph and the weights of each edge in the graph. [1] . We then compute the ASCOS ++ score using the algorithm to get the similarity score matrix for our graph. In the algorithm, it takes the adjacency matrix for the graph as the input. It initializes a guessing matrix S of size (n) where n is the number of nodes in the graph i.e the number of subjects in the graph. It then computes the a similarity score for each node pair in the graph. The similarity score for node i and node j is explained extensively in Chen's work [1]. The similarity score would be 1 if node i = node j The algorithm repeated the above procedure until the similarity score between the previous iteration converges with the current iteration. So we used the above algorithm to calculate the ASCOS ++ similarity matrix of size 40 *40 score of the graph generated in part 1. We assumed the decay factor as 0.9. From the ASCOS ++ matrix we get the 'm' most significant subjects by taking the summation of similarity score for each subject row. In the sample output shown, we have taken subject-subject similarity for CM model , considered 20 similar subjects to calculate the 10 significant subject. The 10 most Significant Subjects are
Subject 21
Subject 25
Subject 13
Subject 27
Subject 9
Subject 38
Subject 6
Subject 5

Subject 4
Subject 23

The number of in -neighbours of each subject are (13, 38.0), (21, 37.0), (7, 35.0), (38, 35.0), (40, 34.0), (3, 33.0), (4, 33.0), (27, 33.0), (6, 32.0), (19, 32.0), (25, 32.0), (5, 29.0), (14, 28.0), (9, 26.0), (32, 26.0), (36, 23.0), (18, 22.0), (11, 21.0), (20, 21.0), (35, 21.0), (15, 20.0), (16, 18.0), (23, 17.0), (2, 16.0), (17, 16.0), (28, 15.0), (12, 14.0), (31, 14.0), (29, 12.0), (30, 12.0), (10, 10.0), (39, 10.0), (26, 7.0), (24, 6.0), (33, 6.0), (34, 5.0), (37, 5.0), (8, 3.0), (22, 3.0), (1, 0.0)]
As we can see that subject 21 is the most significant subject from the 20 similar subjects we considered for the graph. Although the Subject 13 has the most number of in-edges but Subject 21 is marked as significant because our ASCOS ++ features considers the weight of the edge along with the number of in neighbour of the nodes. 21 has edges with more weight than 13. Hence it ranks 21 higher than 13.

3 Interface specifications

To get a look at the help message, run:
python ./main.py help

To perform task 1: python ./main.py t1
Followed by providing:
feature model (CM, ELBP, HOG)
specified X value (cc, con, detail, emboss, jitter, neg, noise1, noise2, original, poster, rot, smooth, stipple)
specified k value (1-192 for CM, 1-32 for ELBP, 1-1764 for HOG)
specified dimensionality reduction technique (SVD, PCA)

Two files will be created in the latent semantic folder:
latent_semantics_(dimensional_reduction_technique)_(feature_model)_(type)_(k).txt which has the subject-weight pairs and
rlatent_semantics_(dimensional_reduction_technique)_(feature_model)_(type)_(k).txt which has k by f matrix for projection.

To perform task 2: python ./main.py t2
Followed by providing:
feature model (CM, ELBP, HOG)
specified Y value (1-40)
specified k value (1-192 for CM, 1-32 for ELBP, 1-1764 for HOG)
specified dimensionality reduction technique (SVD, PCA)

Two files will be created in the latent semantic folder:
latent_semantics_(dimensional_reduction_technique)_(feature_model)_(subject_number)_(k).txt which has the subject-weight pairs and
rlatent_semantics_(dimensional_reduction_technique)_(feature_model)_(subject_number)_(k).txt which has k by f matrix for projection.

To perform task 3: python ./main.py t3
Followed by providing:
feature model (CM, ELBP, HOG)
specified k value (1-12)
specified dimensionality reduction technique (SVD, PCA)

input directory (database)

Two files will be created:
latent_semantics_(type_type)_dimentionality_reduction_feature_model_k.txt which will have the type-weight pairs and is stored in the latent semantic folder and
(type_type)_feature_model.txt which has the similarity matrix for the types and is stored in the sim_matrix folder.

To perform task 4: python ./main.py t4
Followed by providing:
feature model (CM, ELBP, HOG)
specified k value (1-40)
specified dimensionality reduction technique (SVD, PCA)
input directory (database)

Two files will be created:
latent_semantics_(subject_subject)_dimentionality_reduction_feature_model_k.txt which will have the type-weight pairs and is stored in the latent semantic folder and
(subject_subject)_feature_model.txt which has the similarity matrix for the types and is stored in the sim_matrix folder.
Just use database as the directory name to avoid any issues.

To perform task 5: python ./main.py t5
Followed by providing:
folder name (images,database)
file name
latent semantic file name
number of images to return

Place a new image in images and then use that directory for the folder name. If using an image in the database use database as the image name.
Image names and latent semantic files should include the .png and .txt file endings.
For the latent semantic files just use the name without the r in front. For example use latent_semantics_PCA_CM_cc_20.txt even though the real file will be that with an r in front.
These latent semantic files should be files generated from tasks 1 and 2.

To perform task 6: python ./main.py t6
Followed by providing:
folder name (images,database)
file name
latent semantic file name

Place a new image in images and then use that directory for the folder name. If using an image in the database use database as the image name.
Image names and latent semantic files should include the .png and .txt file endings.
For the latent semantic files just use the name without the r in front. For example use latent_semantics_PCA_CM_cc_20.txt even though the real file will be that with an r in front.
These latent semantic files should be files generated from tasks 1 and 2.

To perform task 7: python ./main.py t7

Followed by providing:
folder name (images,database)
file name
latent semantic file name

Place a new image in images and then use that directory for the folder name. If using an image in the database use database as the image name.
Image names and latent semantic files should include the .png and .txt file endings.
For the latent semantic files just use the name without the r in front. For example use latent_semantics_PCA_CM_cc_20.txt even though the real file will be that with an r in front.
These latent semantic files should be files generated from tasks 1 and 2.

To perform task 8:
python ./main.py t8
Followed by providing
matrix file name
n value
m value

The matrix file name should come from the sim_matrix folder and should include the file ending.

4 System requirements/installation and execution instructions

The program is designed to run in Python version - 3.7.6. Once the version or above is installed, use pip installer to add the following libraries:
—-numpy
—-Sklearn
—-matplotlib
—-skimage
—-pandas
—-scipy
—-Networkx

We have following folders and files:

–Database - It contains all the images used for the projects

–Latent_semantics_files - Stores the latent semantics for different feature models and different dimensionality reduction techniques which we got from task 1 and 2

–Output : Contains the output for task 8

–Python files for task 1 to task 8

–main.py - We run our program from main.py file which performs each task. More information on running the individual tasks is shared in the interface specifications.

–Sim_matrix - Stores the similarity matrix obtained from task 3 and task 4 and used a input file for task 8

–Images - It is used to store the query images which are not in database which is used as an input for task 5,task 6 and task 7

5 Related Works

In our project we are experimenting on graph analysis and dimensionality reduction techniques like Principal Component Analysis and Singular Value Decomposition. Mohammed et al [2017] [2] have explored PCA as a dimensionality reduction method. PCA was applied to image features extracted using Scale Invariant Feature Transform (SIFT) features and Speeded Up Robust Features (SURF). After comparing multiple sets of experimental data with different image databases PCA with a reduction in the range, can effectively reduce the computational cost of image features, and maintain the high retrieval performance as well.

Cha, G.-H. (2016) [9] proposed Kernel PCA as a nonlinear extension of the popular PCA. The basic idea is to first map the input space into a feature space via a nonlinear map and then compute the principal components in that feature space. It illustrates the potential of kernel PCA for dimensionality reduction and feature extraction in multimedia retrieval. Extensive experimental results show that kernel PCA performs better than linear PCA with respect to the retrieval quality as well as the retrieval precision in content-based image retrievals.

SVD is a linear algebra technique used for discovering correlations within data. Wilson et al [2011] [8] referred SVD as query quality refinement (QQR) technique, improves the image similarity search result, and when incorporated with genetic algorithms further optimizes the search. The focus of this work is the application of image enhancement technique, using histogram equalization, to the images retrieved using singular value decomposition. These beneficial applications can be extended to other different types of multimedia data in various areas such as the P2P and WiMAX networks.

Lang et al [2018] [6] have proposed an innovative fast graph search algorithm named LSH-GSS, which first transforms complex graphs into vectorial representations based on prototypes in the database and later accelerates a query in Euclidean space by employing LSH because images can be represented as attributed graphs. Attributed graphs are transformed into n-dimensional vectors and apply LSH-GSS to execute further image retrieval.

Similarity measure is a crucial step in structural classification. In many applications, this step results in a sub-graph isomorphism problem. This problem is known in graph theory by a combinatorial explosion. To get around this problem, Ali et al [2012] [3] have proposed a new approach which considers a graph as a set of paths that compose it. The matching, paths allows reducing the combinatorial cost.

6 Conclusion

In this project we were able to take a database of images, calculate the features of the images, and then reduce the dimensions to features using latent semantic features. We were also able to project image vectors, type vectors, and subject vectors into the new feature space and calculate the similarity between the various groups. We also worked with creating similarity matrices and creating graphs that represent the similarities between those same groups. Tasks 5 through 8 were the most interesting and testable tasks. The results for tasks 5 and 6 were both very positive with similar images and types being returned very frequently while task 7 was not very good at returning the subjects of the images. Task 8 also provided positive results as it seemed to accurately relate the importance of the similarities between subjects. There was a lot of struggles and interesting results along the way, but the end product seems to accomplish all of the major goals.

Bibliography

[1] Hung-Hsuan Chen and C. Lee Giles. “ASCOS++: An Asymmetric Similarity Measure for Weighted Networks to Address the Problem of SimRank”. In: *ACM Trans. Knowl. Discov. Data* 10.2 (Oct. 2015). ISSN: 1556-4681. DOI: `10.1145/2776894`. URL: `https://doi.org/10.1145/2776894`.

[2] N. Dalal and B. Triggs. “Histograms of oriented gradients for human detection”. In: 1 (2005), 886–893 vol. 1. DOI: `10.1109/CVPR.2005.177`.

[3] Bruce A. Draper et al. “Recognizing faces with PCA and ICA”. In: *Computer Vision and Image Understanding* 91.1 (2003). Special Issue on Face Recognition, pp. 115–137. ISSN: 1077-3142. DOI: `https://doi.org/10.1016/S1077-3142(03)00077-8`. URL: `https://www.sciencedirect.com/science/article/pii/S1077314203000778`.

[4] Xiaofeng Fu and Wei Wei. “Centralized Binary Patterns Embedded with Image Euclidean Distance for Facial Expression Recognition”. In: *2008 Fourth International Conference on Natural Computation*. Vol. 4. 2008, pp. 115–119. DOI: `10.1109/ICNC.2008.94`.

[5] Abir Ghosh et al. “Grid Color Moment Features in Glaucoma Classification”. In: *International Journal of Advanced Computer Science and Applications* 6 (Sept. 2015). DOI: `10.14569/IJACSA.2015.060913`.

[6] Charles R. Harris et al. “Array programming with NumPy”. In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: `10.1038/s41586-020-2649-2`. URL: `https://doi.org/10.1038/s41586-020-2649-2`.

[7] Andreas Höcker and Vakhtang Kartvelishvili. “SVD approach to data unfolding”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 372.3 (1996), pp. 469–481. ISSN: 0168-9002. DOI: `https://doi.org/10.1016/0168-9002(95)01478-0`. URL: `https://www.sciencedirect.com/science/article/pii/0168900295014780`.

[8] Priti Maheshwary and Namita Srivastava. “Prototype System for Retrieval of Remote Sensing Images based on Color Moment and Gray Level Co-Occurrence Matrix”. In: *P. Maheshwary and N. Srivastava, "Prototype System for Retrieval of Remote Sensing Images based on Color Moment and Gray Level Co-Occurrence Matrix", International Journal of Computer Science Issues, IJCSI, Volume 3, pp20-23, August 2009* 3 (Aug. 2009). URL: `http://cogprints.org/6702/`.

[9] M. D. Malkauthekar. “Analysis of euclidean distance and Manhattan Distance measure in face recognition”. In: *Third International Conference on Computational Intelligence and Information Technology (CIIT 2013)*. 2013, pp. 503–507. DOI: `10.1049/cp.2013.2636`.

[10] L. Mdakane and F. V. D. Bergh. “Extended local binary pattern features for improving settlement type classification of quickbird images”. In: (2012).

[11] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[12] M. Pietikäinen. “Local Binary Patterns”. In: *Scholarpedia* 5.3 (2010). revision #188481, p. 9775. DOI: `10.4249/scholarpedia.9775`.

[13] Stéfan van der Walt et al. “scikit-image: image processing in Python”. In: *PeerJ* 2 (June 2014), e453. ISSN: 2167-8359. DOI: `10.7717/peerj.453`. URL: `https://doi.org/10.7717/peerj.453`.

7 Appendix

] It is important to not that LDA, K-means, and task 9 were all approved to be dropped as one of our members dropped the course and left us with only five members instead of 6.

Tanner Greenhagen- Was the lead architect on PCA and SVD and wrote a lot of the feature utilities that were used for the rest of the tasks. His focus was on tasks 1 and 2.
Khushank Goyal- Implemented tasks 3 and 4 for creating and storing the similarity matrix and using dimensionality reduction on the similarity matrix.
Varad Joshi - Implemented task 8 which takes the subject-subject similarity matrix and creates a graph with n most similar subjects for each subject and performs ASCSO ++ algorithm to get m significant images and command line input for all tasks.
Connor Wardell - Implemented SVD and tasks 6 and 7 which takes in a query image, a latent semantics file and either a subject label y or type label x and then returns a either subject or a type that is associated with the query image.
Mounish Rayudu Chennupati - Task 5 implementation which takes in a query image, a latent semantics file then it will identify and visualise most similar n images under the selected latent semantics.