Analysis of the 11k Hands dataset

Group Members

1. Dhruv Patel / dpatel49@asu.edu
2. Harsh Virani / hvirani@asu.edu
3. Jaykumar Vaghasiya / jvaghasi@asu.edu
4. Satyak Patel / smpate12@asu.edu
5. Tarun Chinmai Sekar / tsekar@asu.edu
6. Utkarsh Agarwal / uagarwa5@asu.edu

Abstract

The 11khands dataset is a collection of 11076 hand images from a collection of 190 subjects. The dataset features hands in both the dorsal and palmar positions against a uniform white background.

This phase of the project involved a lot of experimentation, enabling us to fine tune the parameters to get the desired results. The experimentation helped in learning of clustering,indexing and classification of various images.For the first 2 tasks classification of dorsal and palmer images were done using by getting k(user given)- latent semantics and clustering respectively. For 30 latent semantics using SVD we got 62% accuracy. Visualisation provided for task 3 gives K most dominant images for 3 userIds. This is measured via the Personalised Page Rank (PPR) algorithm.Three models namely PPR, decision tree and SVM were implemented from scratch and were subsequently trained on training images having dorsal/palmar images. After careful experimentation , decision trees perform best on most datasets, but PPR may perform better under some conditions as well.Furthermore, the LSH algorithm is devised which will index the images according to its similarity with the other images and make buckets for each images with similar images having the closest distances. On query it will give the best t images from the bucket of query image. Finally there is a relevant feedback system and probabilistic relevance feedback system which takes in user inputs for relevance and rearranges/ reprocesses to get better classifications. We must note the relevant feedback is an added extra layer to the already existing classifiers to get better results.

Keywords
Local Binary Patterns, Color Moments, Feature Extraction, 11kHand, Locality Sensitive Hashing.

Introduction

The aim of this project is to learn about the feature extraction techniques on Multimedia. Furthermore, we have learnt about different dimensionality reduction techniques like PCA, SVD, NMF, LDA. To find k nearest neighbors we learnt LSH (Local Sensitive Hashing) which is similar to k nearest neighbor search but less computationally expensive. We created SVM, Decision Tree classifier to classify images as dorsal or palmar and relevant or irrelevant. We learnt about PPR (Personalized Page Rank) Algorithm and implemented it to compute page rank for given set of images and using the page ranks to sort and classify images.

Terminology

1. Personalized Page Rank

Pagerank is the algorithm that is used to measure the quality or importance of a webpage. Pagerank of a particular webpage is computed by using the count of web pages pointing to that web page and the count of outgoing links from each of those web pages. Basically, Pagerank can be interpreted by considering a user visiting a particular webpage. By some probability $1-\beta$ the user makes a jump to any random webpage and with probability β he goes to one of the web pages being pointed by the current webpage.
Personalized Pagerank is similar to Pagerank however, the difference being in the random jump being made by the user. In Personalized Pagerank, the jump being made with probability $1-\beta$ is to a particular set of webpages based on the user preference.

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

Where T is the Transition Matrix, β is the probability with which user will jump to one of the node in the outbound set of the current node, $PR(t)$ is the set of Page Rank of all nodes at iteration t, s is the seed matrix.

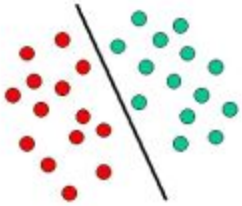
2. Decision Tree

Decision Tree is a naive supervised classification algorithm where the data is split into multiple datasets based on the if conditions at every level.So,basically at every step the dataset encounters an if condition and based on the if condition the dataset splits into two buckets and this goes on recursively till we reach all the leaves which has predictions.The goal of the algorithm is to unmix the labels as we proceed down the tree. In other words,produce the purest possible distribution at each node.
When we want to classify any new data , we would just pass it to the tree. The tree would go down and apply all kinds of rules at every stage and once done , it will make it reach any of the final leaves. As the leaves contain unmixed labels , we can rightly predict the correct label for that test row.

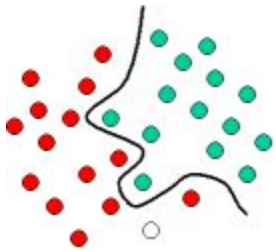
3. Support Vector Machine - SVM

Support vector machines is a machine learning algorithm which analyzes data for classification and regression analysis. It is a supervised learning method that classifies data into one of two categories. An SVM outputs a map of the sorted data with the margins between the two as far apart as possible. SVMs are used in various fields such as text categorization, image classification, handwriting recognition and in the sciences. Similar to other supervised learning machines, SVM requires labeled data to be trained. SVM is trained with a series of data already classified into two categories, building the model as it is initially trained. SVM algorithm then determines which category a new data point belongs in.

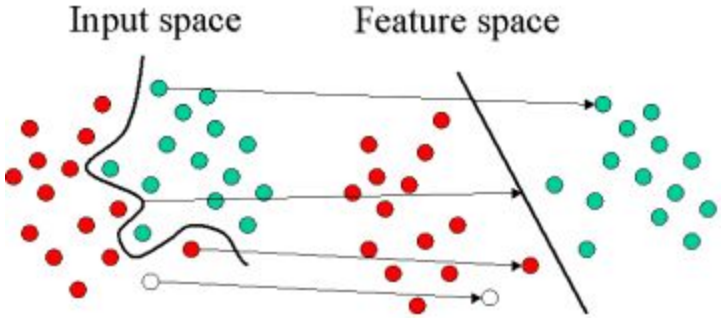
Support Vector Machines are based on the concept of decision planes that define decision boundaries. A decision plane is one that separates between a set of objects having different class labels. In the figure below, the objects belong either to class green or red. The separating line defines a boundary on the right side of which all objects are green and to the left of which all objects are red. Any new object (white circle) falling to the right is labeled as green as red should it fall to the left of the separating line).



The above is an example of a linear classifier which separates given objects into their respective groups with a line. Most classification tasks, however, are not that simple, and generally more complex structures are needed in order to make an optimal separation, i.e., correctly classify new objects on the basis of the available training data. This situation is depicted in the below figure. Compared to the previous schematic, it is clear that a full separation of the green and red objects would require a curve (which is more complex than a line). Classification tasks based on drawing separating lines to distinguish between objects of different class memberships are known as hyperplane classifiers. Support Vector Machines are particularly suited to handle such tasks.



The figure below shows the basic idea behind SVM. Using a set of mathematical functions, known as kernels, we see the original objects mapped, i.e., rearranged, . The process of rearranging the objects is known as mapping (transformation). Note that in this new setting, the mapped objects (right side of the schematic) is linearly separable and, thus, instead of constructing the complex curve (left schematic), all we have to do is to find an optimal line that can separate the green and the red objects.



Kernel Functions:

$$K(\mathbf{X}_i, \mathbf{X}_j) = \begin{cases} \mathbf{X}_i \cdot \mathbf{X}_j & \text{Linear} \\ (\gamma \mathbf{X}_i \cdot \mathbf{X}_j + C)^d & \text{Polynomial} \\ \exp(-\gamma |\mathbf{X}_i - \mathbf{X}_j|^2) & \text{RBF} \\ \tanh(\gamma \mathbf{X}_i \cdot \mathbf{X}_j + C) & \text{Sigmoid} \end{cases}$$

where $K(\mathbf{X_i}, \mathbf{X_j}) = \phi(\mathbf{X_i}) \bullet \phi(\mathbf{X_j})$

The kernel function, represents a dot product of input data points mapped into the higher dimensional feature space by transformation ϕ. Gamma is an adjustable parameter of certain kernel functions.

The RBF is by far the most popular choice of kernel types used in Support Vector Machines. This is mainly because of their localized and finite responses across the entire range of the real x-axis. RBF kernel works well in practice and it is relatively easy to tune.

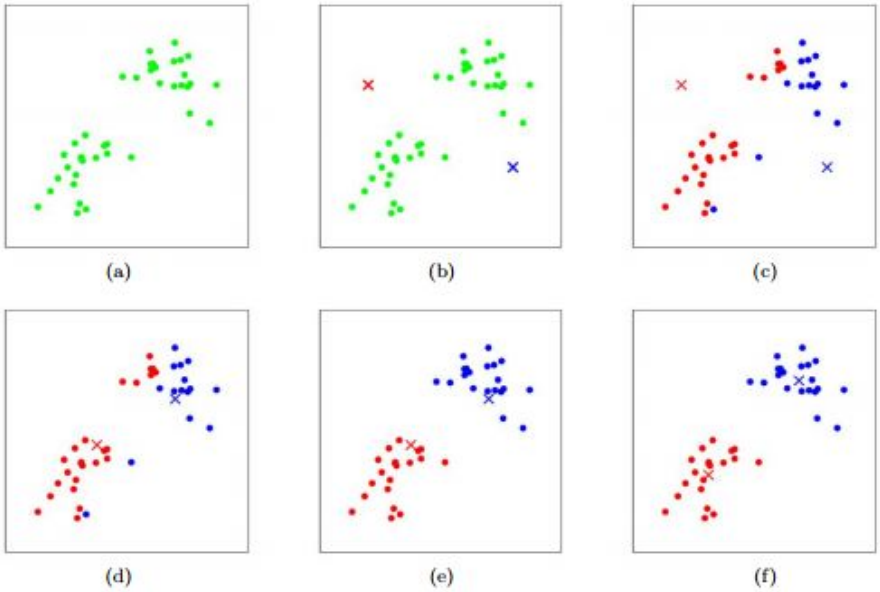
4. K-means Clustering

K-means Clustering is the simplest unsupervised machine learning algorithm. The algorithm works by aggregating similar data points. The similarity metric can be as simple as Manhattan/ Euclidean etc. The K-means algorithm identifies K number of centroids and then allocates the data points to the nearest clusters. The algorithm starts with a randomly initiated centroids. These centroids are used as the starting points for every cluster. The algorithm then performs the iterative optimization steps to update the position of the clusters. The algorithm stops when the movement of clusters have stabilized.

The working structure of the algorithm can be defined as follows:

1. Randomly initialize k centroids.
2. We categorize each item to its closest centroid.
3. Once categorized, we update the centroid's coordinates by averaging the data points in that cluster.
4. Repeat this process until the positions of clusters stop changing.

Figure below shows the iterative process.



5. Singular Value Decomposition(SVD)

Singular value decomposition is commonly used dimensionality reduction approach in exploratory data analysis and machine learning. Singular value decomposition is factorization of rectangular matrix into three matrices of given form,

$$A_{m\times n} = U_{m\times m} S_{m\times n} V^T_{n\times n}$$

Where, U and V^T are orthogonal matrices and S is a diagonal matrix.

$$U^T U = V V^T = I$$

Here, the columns of U are left singular vectors and the rows of V^T are the right singular vectors. S has singular values and is diagonal. The SVD represents and expansion of the original data in a coordinate system where the covariance matrix is diagonal.

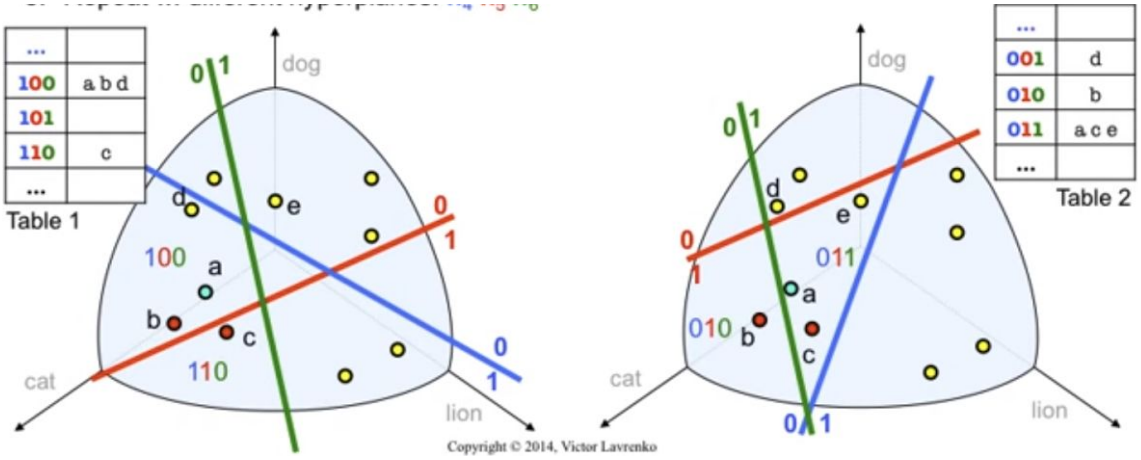
To calculate SVD, finding the eigenvalues and eigenvectors of AA^T and A^TA is required. The eigenvectors of A^TA make up the column of V and the eigenvector of AA^T make up the columns of U. The singular values in S are square roots of eigenvalues from AA^T or A^TA. The singular values are diagonal entries of the S matrix and are arranged in descending order. The singular values are always real number is our original matrix is real matrix.

The sum of squares of the singular values should be equal to the total variance in A. So, the size of each tells you how much of the total variance is accounted for by each singular vector. A typical machine learning problem might have several hundred or more variables. This makes singular value decomposition indispensable in machine learning or any other cases where variable reduction is required.

6. Locality Sensitive Hashing (LSH)

Locality Sensitive Hashing can be considered as a K Nearest Neighbour search with the main difference being less computationally expensive. While in KNN, we look for similar items by comparing every data point, in LSH we only compare the datapoints within the same bucket. The buckets are initialized in a way that each bucket will contain only the spatially similar data points. LSH generates hashcodes for the data points in a way that similar data points will have the same hashcodes. It essentially labels a subspace of the multidimensional vector space by a code. While querying, we use this hashcode as the index. This operation is in the order of O(1) (constant). This index points to the bucket which contains spatially similar data-points. We need to apply the search on only these small fractions of data-points.

In order to remove false-positives, LSH partitions the space in L different manners. It essentially means that the a vector space has L hash tables. This is because, by using different random partitionings we avoid the mistake of similar data being put into different buckets by the randomly initialized partition vector. LSH returns all the data points that are common across all the hash table buckets.



Problem Specification

This phase of the project comprises 6 distinct tasks. These tasks are defined as follows.

Task 1

a. Inputs: A folder of images with dorsal and palmar labels, a folder with a set of unlabelled images.
b. Compute “k” latent semantics of “dorsal” images and “k” latent semantics of “palmar” images. Using only the latent semantics of these two labelled groups, classify the unlabelled data into “dorsal” and “palmar” images.
c. Output: Display the results of the classification.

Task 2

a. Inputs: A folder of images with dorsal and palmar labels, a folder with a set of unlabelled images, a value “c”.
b. Compute “c” clusters for “dorsal” images, and “c” clusters for “palmar” images. Using only the descriptions of the clusters, classify the unlabelled data into “dorsal” and “palmar” images.
c. Output: Display the results of the classification.

Task 3

a. Inputs: A value “k”, another value “K”, three image IDs,
b. Using the user supplied “k”, create an image-image similarity graph, with each image having “k” outgoing edges to “k” similar images. Visualize the “K” most dominant images using Personalized Pagerank.
c. Output: Visualizations of the “K” most dominant images compared to the user specified images.

Task 4

a. Inputs: Choice of classifier - SVM, Decision Tree or Personalized Pagerank, A folder with labelled images, A folder with unlabelled images.
b. Train the classifier chosen by the user on the labelled images and then use the trained classifier to predict the unlabelled images
c. Output: The classification results

Task 5a:

a) Inputs: Number of Layers (L), number of Hashes (k), the set of input vectors
b) Create an index of the input vectors using a locality sensitive hashing tool with for euclidean distance.
c) Outputs: An index structure built using LSH

Task 5b:

a) Inputs: Query image id, and a value “t”
b) Using the LSH index, find and visualize top “t” images that are similar to the query image.
c) Outputs: Visualize the top “t” similar images for the given query image

Task 6

a) Inputs: Results of top “t” similar images for a given query, a set of “relevant” and “irrelevant” images, choice of the classifier - SVM, PPR, Decision Tree, Probabilistic Relevance Feedback
b) Using the feedback provided, train the classifier chosen by the user, and use it to re-order the results of the “t” similar images query.
c) Output: An iterative improvement of the “t” similar images.

Assumptions

- We are using Euclidean distance as a similarity measure. We considered that metric assuming it will provide good enough results.
- The metadata for all images in the dataset is available.
- The provided directory contains only images, and no other files.
- The images are in the RGB space, and are of similar size and dimensions.
- Only hand images are present in the dataset.
- The latent semantics are “enough” to describe a image
- The reduced latent semantics are present in the database for task 5a.
- Color moments are a good mechanism to discriminate between dorsal and palmar images.
- In pagerank, the number of outgoing edges for each image will always be less than the total number of images or nodes in the graph.

Description of the proposed solution/implementation

The proposed solution uses mongoDB as the backend for storing intermediate results during processing. The metadata for the given image dataset needs to be supplied to the implementation in the form of a csv file. The image features along with the metadata information is stored in mongodb in the form of a document, with the following schema.

{

“image_path” : string, #Path to the image on the disk

“**HOG” : array(Float) #**The array of floating point numbers representing the Histogram of Oriented Gradients for the image.

“**HOG_reduced” : array(Float) #**The HOG after dimensionality reduction to “k” features.

“id”: int, # ID of the subject whose hand is photographed.

“age”: int , # Age of the subject.

“Gender”: string, # Gender of the subject (Values: male/ female).

“skinColor”: string, # Skin Color of the subject (Values: fair/ dark/ medium/ very fair).

“Accessories”: int, # Indicates if the subject whose hand is photographed is wearing any accessories on hand. (Values: 0 - No accessories, 1 - Wearing some accessory)**.**

“naiPolish”: int, # Indicates if the subject whose hand is photographed has nailPolish on his/her fingernails. (Values: 0 - No nail polish, 1 - having nail polish)**.**

“aspectOfHand”: string, # Aspect of the hand image indicating if it is Dorsal/ Palmer and Left/Right

“Irregularities”: # Indicates if the subject whose hand is photographed has any irregularities on hand. (Values: 0 - No irregularities , 1 - having irregularities)**.**

“imageName”: string, # Filename of the image.

}

Description of classes utilized by all 6 tasks

All helper functions are segregated in four classes named feature_extraction, dimensionality_reduction, distance_measure and project_utils.

Feature Extraction Class Methods

- **Extract_hog_features**

  Input: Path to the image
  Output: Feature vector
  Description: Image is downsampled by a factor of 1 per 10. HOG feature vector is calculated using the *sklearn* library. Parameter is given as 9 for orientation, 8x8 for pixels per cell, 2x2 cells per block. This vector is stored in mongodb database where document id is image ID.

Dimensionality Reduction Class Methods

- **Reduce_dimensions_svd**

  Input: Data matrix, k
  Output: Reduced data matrix
  Description: U, S and V^T are calculated using *numpy* library. All three matrices are sliced according to given k. Reduced matrix is calculated as matrix multiplication of U and S. Reduced data matrix is returned in the end.

- **Reduce_dimensions_lda**

  Input: Data matrix, k
  Output: Reduced data matrix
  Description: *Sklearn* library is used for calculating LDA of data matrix. K is used as n_components and 0 is used as a random state for the parameter of LDA function of Sklearn. Reduced data matrix is returned in the end.

Task Description

Task 1:
The metadata for the labelled images are first processed, and inserted into the database. The CM features for all the images are first extracted. Then, dimensionality reduction is performed using the SVD algorithm. This algorithm was chosen as it had the most discriminating power amongst PCA, LDA and

NMF. The value of “k” was chosen to be 20 after multiple experiments. The dorsal and palmar feature latent matrices are created and stored separately in memory.

The testing folder is enumerated and CM is extracted for each image in the test folder. The dot product of the CM feature vector with the palmar feature matrix and the dorsal feature matrix is computed. The image is classified to the class with the higher dot product. The accuracy obtained by this method is tabulated in the outputs section.

Task 2

In this task given as input a folder of images labelled as dorsal or palmar and a number c, the program needs to compute c clusters of dorsal images and c clusters of palmar images and visualize the clusters created. And, given input folder of unlabelled images the program label it as dorsal or palmar using the descriptors of the clusters.

For creating the dorsal and palmar images k means algorithm was used. The input image is labelled dorsal or palmar by calculating its distance from the centroids of both the group of clusters and assigning it as one with the shortest distance.

Task 3:

Task 3 computes the Personalized Pagerank of set of images in the given directory by creating a similarity graph which will have 'k' outgoing edges from each image to 'k' most similar images in that directory. After that, it visualizes the 'K' most dominant images based on the computed PPR (Personalized Pagerank) score. The input provided will be the path to image directory, value of 'k' i.e. the number of outgoing edges for the similarity graph, the value of 'K' i.e. the number of dominant images to visualize and the 3 user image ids. First the program will check whether the features for the images in the given directory are present in the database or not. If not, it will populate the database. Then it will create an image image similarity graph based on the Euclidean Distance and HOG features and select only 'k' similar for each image. This graph is then converted to a transition matrix. The transition matrix will have 0 for each pair that does not have an edge between them. For each image, the value in its outbound images will have the value equal to $\frac{1}{out\ degree\ of\ that\ image}$. Then the seed vector will be created which will have all the values as 0 except for the images which are provided as user images in the input. The value for those images will be

$\frac{1}{number\ of\ user\ images}$. The value of β is set to 0.85. Using the above mentioned formula, the pagerank score for each image is computed. The number of iterations is set to 60. Images are sorted based on the decreasing order of personalized pagerank score. After that, top 'K' images from the sorted array are being visualized.

Steps:

1. Create image image similarity graph using Euclidean Distance and HOG features.
2. Select k most similar images for each image based on the score and create Transition Matrix.
3. Create seed vector using the user images provided as input.
4. Initialize Pagerank vector with $\frac{1}{total\ number\ of\ images}$ as value for all images.
5. Compute Pagerank at each iteration using the old Page Rank values. This is done for 60 iterations.
6. Resulting Page Rank vector is sorted by maintaining the index of images.
7. Top K images are visualized.

Task 4:

This task requires us to build three classifiers based on SVM, Decision-tree and PPR. Then given a folder of unlabeled images, the program labels the images as dorsal-hand or palmar hand using one of the three classifiers selected by the user.

Initially we fetch the Color Moments features of the given image set and create two data matrix using dorsal and palmar as filter string. Color moments are chosen as they highly discriminate between the two classes - dorsal and palmar. Two separate lists of labels, dorsal and palmar are created of size of the corresponding data matrix, of which dorsal values are stored as 1 and plamar values are stored as 0. Both the labeled data lists and the feature data matrices are then concatenated separately. The dimensions of the features in the concatenated feature data matrix is then reduced to 20 using SVD. Then as per the classifier chosen by the user, the model is trained and will be tested against the test images.

Decision Tree based classifier:

Steps:

1. Select the true/false rule which gives maximum information gain or lowers the gini impurity of the dataset at that level of the tree.
[Gini impurity: Helps in quantification of uncertainty at any particular node. It can also be defined as the chance of being incorrect if we randomly assign a label to example in the same set.].
2. To select the best question at each node, we will generate all the possible questions and partition the dataset using those questions. For each question, we will calculate the information gain by subtracting the weighted average gini impurity before partition and after partition. And then select the question which maximally minimises the impurity.
3. As we proceed to the bottom of the tree, use the output dataset of the previous rule as input of the new decision rule. Do this recursively and continue dividing the data until there are no further questions to ask and this is the point we will add a leaf to the tree.
4. After the tree is built, we can just predict that particular testing data, following the tree putting it in the correct bin as per the decision rule.
5. When we reach any leaf node, we can classify that testing data row with the particular label we get in that leaf node.

SVM based classifier:

For SVM based classifier cvxopt library is used for quadratic optimization.
standard form of a QP (following CVXOPT notation):

$$\begin{aligned} \min_{x} \quad & \tfrac{1}{2} x^\top P x + q^\top x \\ \text{subject to} \quad & Gx \preceq h \\ & Ax = b \end{aligned}$$

The CVXOPT QP framework expects a problem of the above form, defined by the parameters {P, q, G, h, A, b}; P and q are required, the others are optional.
power

PPR based Classifier:

PPR classifier tries to label the set of unlabelled images using the pagerank score assigned to them. The same process is followed as mentioned in task 3. The similarity graph is created using a set of labelled and unlabelled images followed by its transition matrix. Then for each class label, a seed vector is formed in which there will be 0 except for the images which belong to that class label. For each class, separate pagerank for all images is calculated. The value of

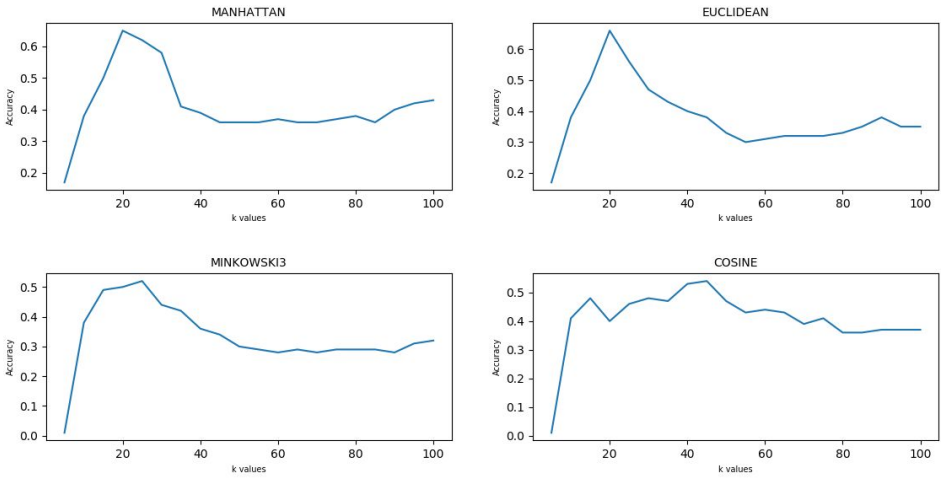
β used is 0.85. Then for each unlabelled image, we label it to the class that has the highest pagerank for that image. If the pagerank of that image is equal for both labels, it doesn't classify that image to either class.

Steps:

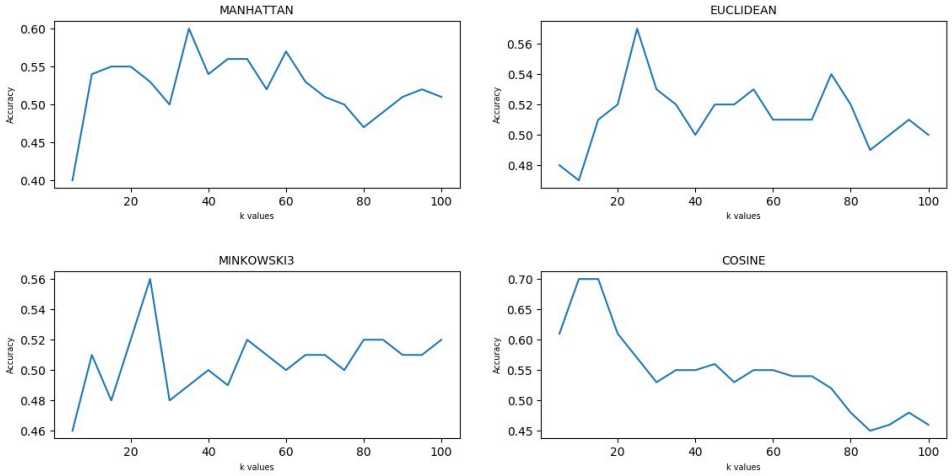
1. Create an image image similarity graph for a set of labelled and unlabelled images using Manhattan Distance and Color Moments as features.
2. Select k most similar images for each image based on the score and create Transition Matrix. The value of k used is 15.
3. Now for each class, create a seed vector.
4. Compute Page Rank for each class.
5. For each image in an unlabelled set, classify it to the class for which its Pagerank is high. If its Pagerank is the same for the class labels, then it doesn't classify that image.

Experiments:

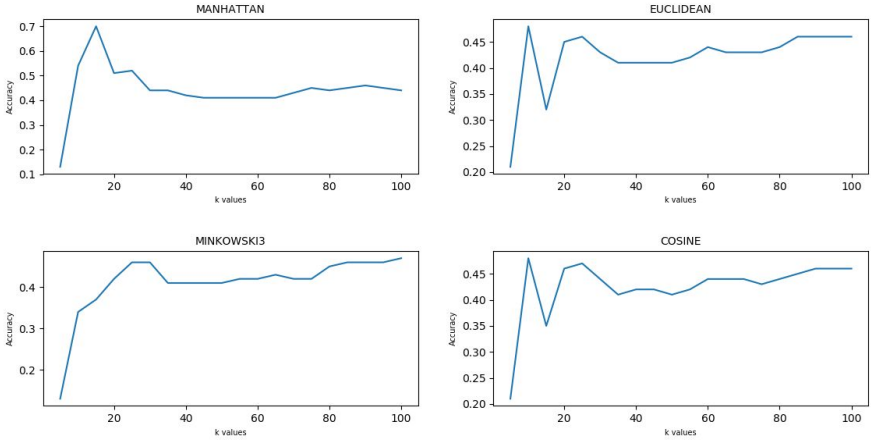
1. HOG, Set1 Labelled, Set1 Unlabelled



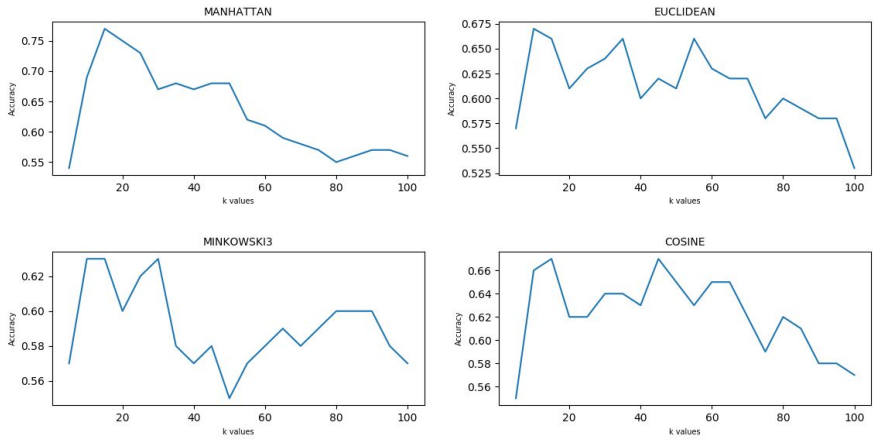
2. HOG, Set2 Labelled, Set2 Unlabelled



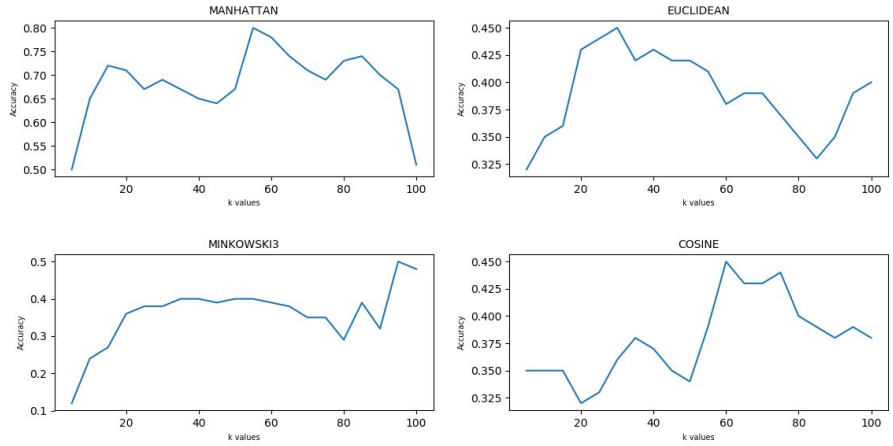
3. CM, Set1 Labelled, Set1 Unlabelled



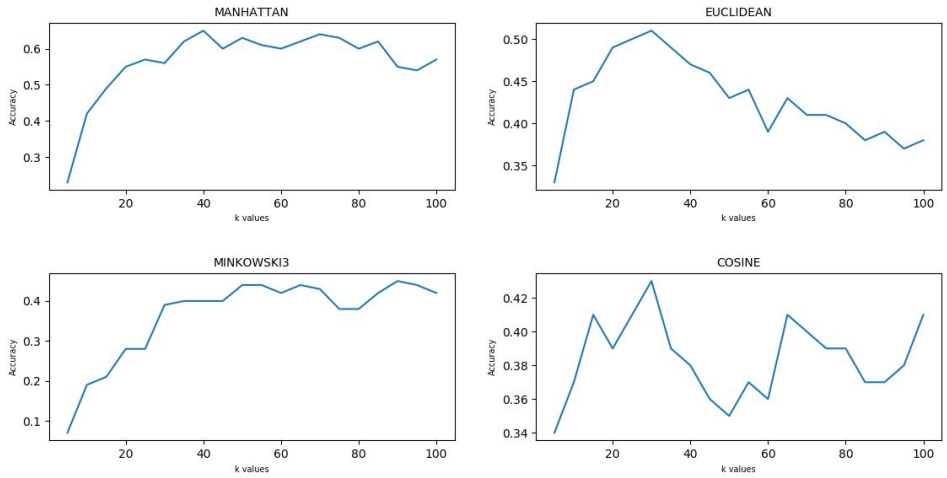
4. CM, Set2 Labelled, Set2 Unlabelled



5. SIFT, Set1 Labelled, Set1 Unlabelled



6. SIFT, Set2 Labelled, Set2 Unlabelled



These graphs show the accuracy for PPR classifier with specified feature extractor and dataset over different values of 'k' and four different distance/similarity measures. As we can see from the graphs, Color Moments with Manhattan gives better accuracy than other combinations for both datasets for the value of k = 15. 'k' represents the number of outgoing edges from each node.

Task 5:

This task asked us to implement a Locality Sensitive Hashing (LSH) where we take the number of hashes, layers and image vectors as input and create an in-memory index structure containing the given set of vectors.

The flow of code 5a is as under:

1. Initialize the random projection vectors.
2. Project the compressed image vectors on the projection vectors. This gives the hash function, essentially the bucket list.
3. Respective images are added into the buckets.
4. Create an index structure by using the number of layers and iterating through the bucket indexes.

For fetching the similar images task 5b:

1. We load the index structure from the MongoDB dataset.
2. Create a list of hash keys for the bucket list by iteratively going deep into the bucket until your layers permit.
3. To find the nearest neighbors, we create a hash for the query vector.
4. Using that hash, we find iteratively deepen into the smaller buckets
5. At the most granular level, we add the image vectors and its distance into the nearest neighbours.

6. Sort the nearest neighbours according to the distances.

Task 6:

Task 6 builds on task 5b, incorporating relevant feedback into the similar image calculation algorithms. The user is asked to choose a number of relevant or irrelevant images, and the feedback given by the user is used to improve the ordering of the results that are returned. The user is initially asked to select a technique for the re-ordering and the iterative process continues until the user is satisfied. The interface displays the images and asks the user to select a number of relevant and irrelevant images. The user's choice is passed to the classifiers, and they are trained based on it.

Relevance feedback using Decision Tree and SVM

1. The images marked as "relevant" and "irrelevant" by the user are labelled as 0 and 1 (-1 and 1 for SVM), and the features of those images are also obtained from the database.
2. The model is then trained to classify the given matrix to "relevant" or "irrelevant".
3. The testing dataset is created by taking all the "t" similar images's features and making them into a matrix.
4. The model is then made to classify the testing data as "relevant" or "irrelevant".
5. The models classifications are iterated, and the distance score is reduced if the image is marked as "relevant" and distance is increased if it is "irrelevant".
6. The images are then sorted by "distance" and presented to the user.
7. The user is then asked to either continue with the iterative process, or exit.

Relevance feedback system for PPR:

In task 6, the inputs are a list of the image given by the output of task 5 and user preference in terms of relevance and irrelevant for some of these images. The functionality of task 6 is to utilize given feedback with help of PPR to reorder the image according to the user preference. To keep track of previous feedback we are adding one more input seed matrix to task 6. So, our function for reordering will take the list of images, feedback and seed matrix (initialized separately for the first time) and return the list of reordered images and updated seed matrix. We will pass this returned seed matrix when providing feedback the next time. In

the next step, the function to create a similarity graph is called first with k and a list of images. This will create a graph with k outgoing edges and images as vertices. Next, the function to create seed matrix is called which will update our seed matrix according to feedback. This function takes an existing seed matrix and adds 1 to the nodes with relevant feedback and subtracts 1 from the nodes with irrelevant feedback. In the end, the seed matrix is normalized and returned. Next, a function for transition matrix is called and it is used to create transition matrix. Next, a function to calculate pagerank is called with a transition matrix and updated seed matrix (according to feedback) to run the page rank algorithm. In the end, images are reordered according to the decreasing order of their PageRank value. The main reordering image function returns the reordered image list and updated seed matrix. This seed matrix can be used for the next iteration. This way we can accumulate all previous feedback through our seed matrix. Here, I'm taking the value of k (no of outgoing edges) as 5, because the value of k has to be less than t (no of images or nodes in the graph). I'm using HOG as a feature extractor for all images in the graph. Euclidean distance is used to find neighbouring nodes in the graph for this task.

Probabilistic Relevance feedback System:

The probabilistic relevance feedback system operates based on probabilities instead of absolute values. It goes through the values of the query vector and counts the occurrences in the entire result set, and in the relevant images set. In order to do the counting, the values are converted into a binary form where values above the mean are considered to be 1 and values below mean are set to 0. After the conversion and the counting, a formula is used to obtain a weight for the given image, and the weight is subtracted from the distance score to re-order the images. The weight is calculated as follows:

N is the total number of values (number of result images * length of feature vector) and R is the total number of "relevant" values (number of relevant images * length of feature vector)

For each relevant images:

1. Get distance of the image from the query vector - di
2. Get count of query vector values in this particular image's vector. This is stored as ni
3. Count occurances of the images's values amongst the relevant images. This is stored as ri
4. Pi is calculated as follows

$$pi = (ri + (ni / N)) / (R + 1)$$

5. Ui is calculated as follows
$$ui = (ni - ri + (ni / N)) / (N - R + 1)$$
6. The weight is calculated as follows
$$weight -= di * log((pi * (1 - ui)) / (ui * (1 - pi)))$$

The reordered images are presented to the user, and the user is asked to either redo the feedback, or exit.

Interface specifications

The solution uses standard python command line parsing utilities to extract the command line arguments passed to the solution such as the path to the image directory, the query image id, the value of “k” etc.
Running the command - python3 task_<taskno>.py -h will display all the necessary parameters needed to be passed for the given task.

The outputs to the given sample queries are present in the Outputs / directory.

System requirements /installation and execution instructions

The solution is designed to work only with Python 3.6+.
The minimum system requirements are:
Processor: 2.2Ghz processor
RAM: 4 GB of RAM
Operating System: macOS or linux (tested only on these platforms)
Python: 3.6

The recommended system requirements are:
Processor: 2.2Ghz processor with at least 4 cores
RAM: 16 GB of RAM
Operating System: macOS or linux
Python: 3.7

The solution also requires the installation of mongoDB, and the creation of a database named “mwdb_project” in the database.
The following python libraries are required to run the program.

1. numpy
2. opencv-python
3. scipy
4. pymongo
5. scikit-image
6. matplotlib
7. opencv-contrib-python-nonfree
8. pandas
9. Scikit-learn
10. cvxopt

To run the solution, the following invocations are used.

Task 1

Python3 task1.py --train_folder ~/Downloads/phase3_sample_data/Labelled/Set1 --train_metadata ~/Downloads/phase3_sample_data/labelled_set1.csv --test_folder ~/Downloads/phase3_sample_data/Unlabelled/Set\ 1 --k 30 --labels_csv ~/ASU/MWDB/Project/DataSet/HandInfo.csv [this parameter is needed only if accuracy calculation is needed]

Task 2

python3 task2.py --train_folder ~/Downloads/phase3_sample_data/Labelled/Set2 --train_metadata ~/Downloads/phase3_sample_data/labelled_set2.csv --test_folder ~/Downloads/phase3_sample_data/Unlabelled/Set\ 1 --c 10 --labels_csv ~/ASU/MWDB/Project/DataSet/HandInfo.csv

Task 3

python3 task3.py --image_folder ~/Downloads/phase3_sample_data/Labelled/Set2 --metadata_file ~/Downloads/phase3_sample_data/labelled_set2.csv --k 5 --K 10 --query_images Hand_0008333.jpg,Hand_0006183.jpg,Hand_0000074.jpg

Task 4

python3 task4.py --train_folder ~/Downloads/phase3_sample_data/Labelled/Set2 --train_metadata ~/Downloads/phase3_sample_data/labelled_set2.csv --test_folder

~/Downloads/phase3_sample_data/Unlabelled/Set\ 2 --classifier PPR --labels_csv ~/ASU/MWDB/Project/DataSet/HandInfo.csv

Task 5a
python3 task5a.py --L 1 --k 4

Task 5b
python3 task5b.py --t 20 --query_image_id Hand_0000674.jpg

Task 6
python3 task6.py --t 20 --query_image_id Hand_0000674.jpg --clf Probabilistic

Outputs:

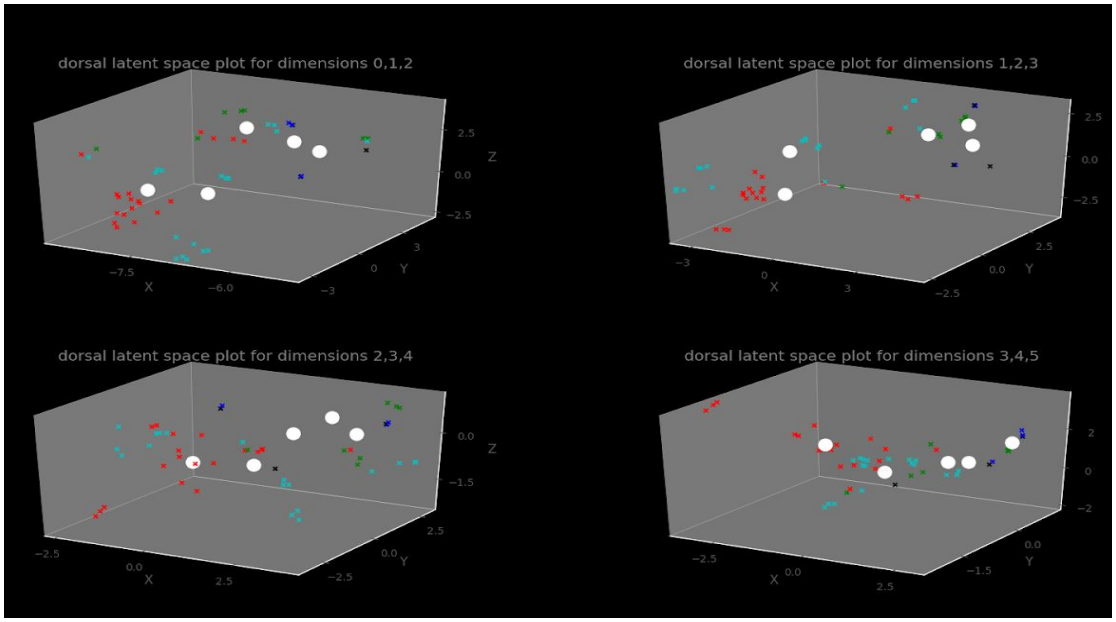
Task 1

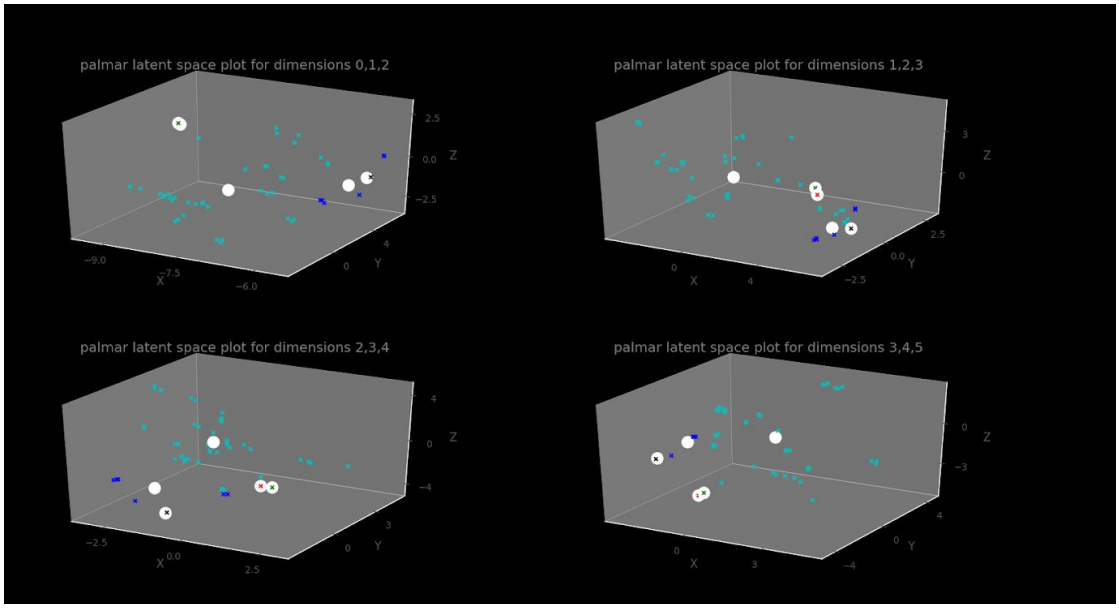
Train Data	Test Data	K	Accuracy
Labelled/Set 1	Unlabelled/Set 1	30	62
Labelled/Set 1	Unlabelled/Set 2	30	59
Labelled/Set 2	Unlabelled/Set 1	30	57.99
Labelled/Set 2	Unlabelled/Set 2	30	60

Task 2

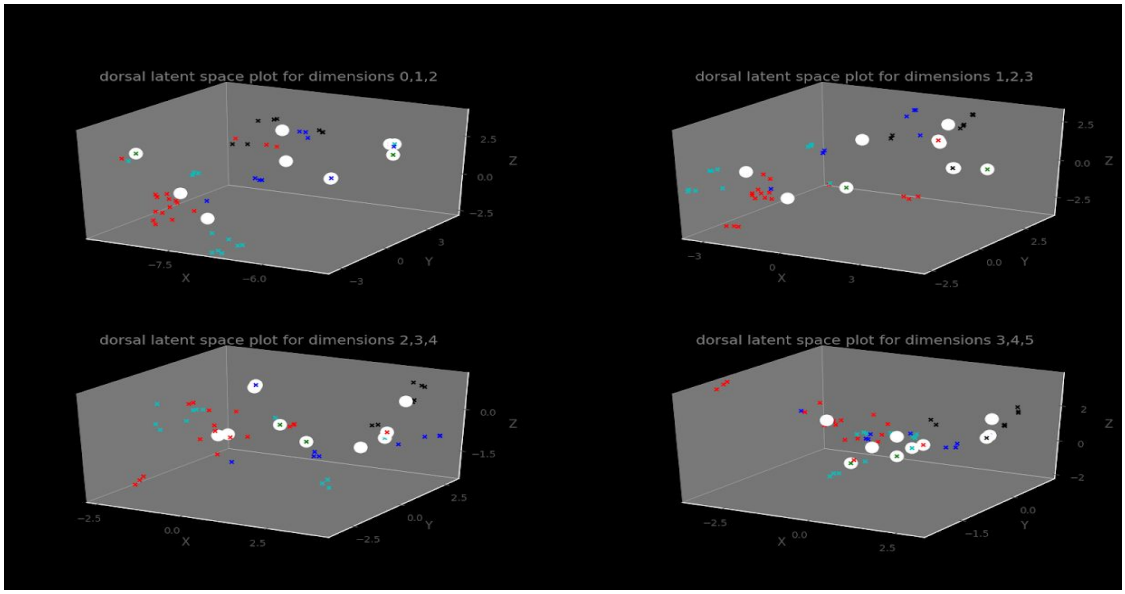
Train Data	Test Data	c	Accuracy
Labelled/Set 2	Unlabelled/Set 1	5	65
Labelled/Set 2	Unlabelled/Set 1	10	68
Labelled/Set 2	Unlabelled/Set 2	5	78
Labelled/Set 2	Unlabelled/Set 2	10	76

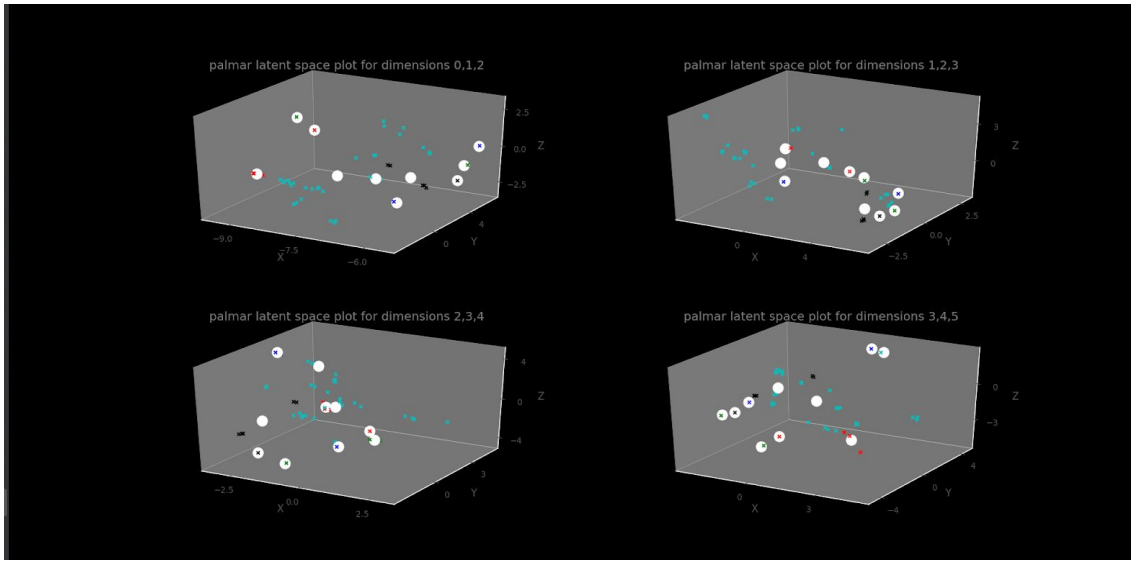
Query-1


dorsal latent space plot for dimensions 0,1,2
dorsal latent space plot for dimensions 1,2,3
dorsal latent space plot for dimensions 2,3,4
dorsal latent space plot for dimensions 3,4,5

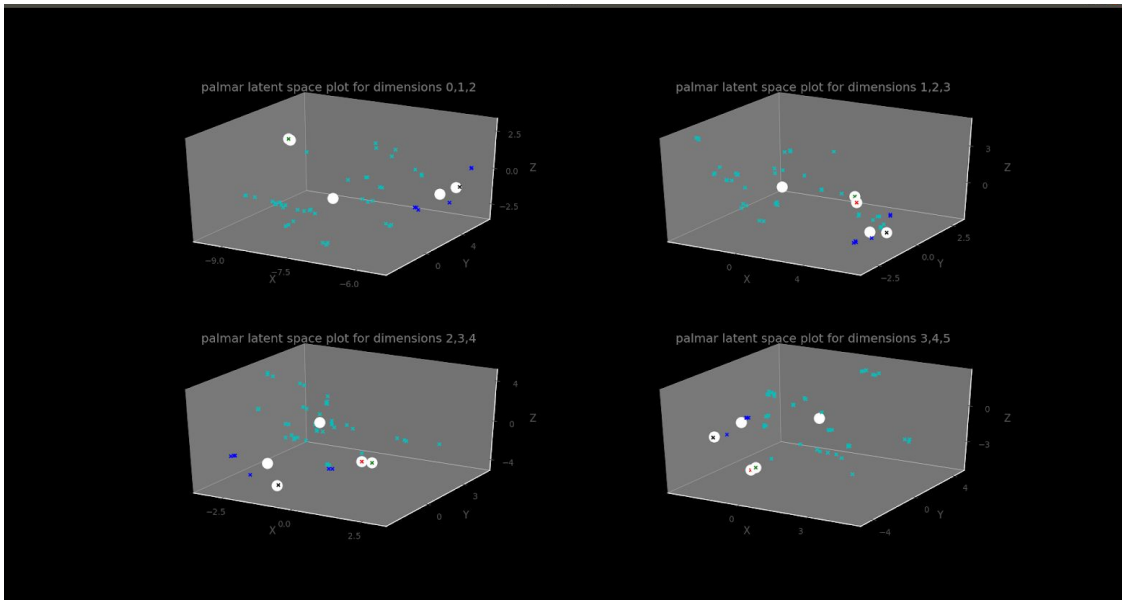

palmar latent space plot for dimensions 0,1,2
palmar latent space plot for dimensions 1,2,3
palmar latent space plot for dimensions 2,3,4
palmar latent space plot for dimensions 3,4,5

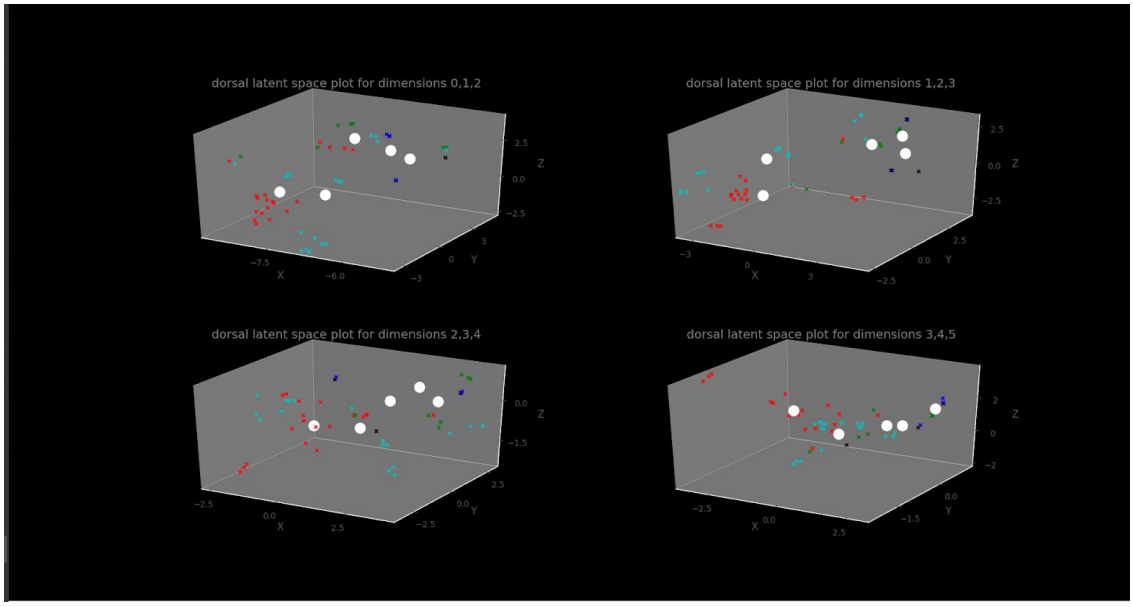
Query 2:


dorsal latent space plot for dimensions 0,1,2
dorsal latent space plot for dimensions 1,2,3
dorsal latent space plot for dimensions 2,3,4
dorsal latent space plot for dimensions 3,4,5

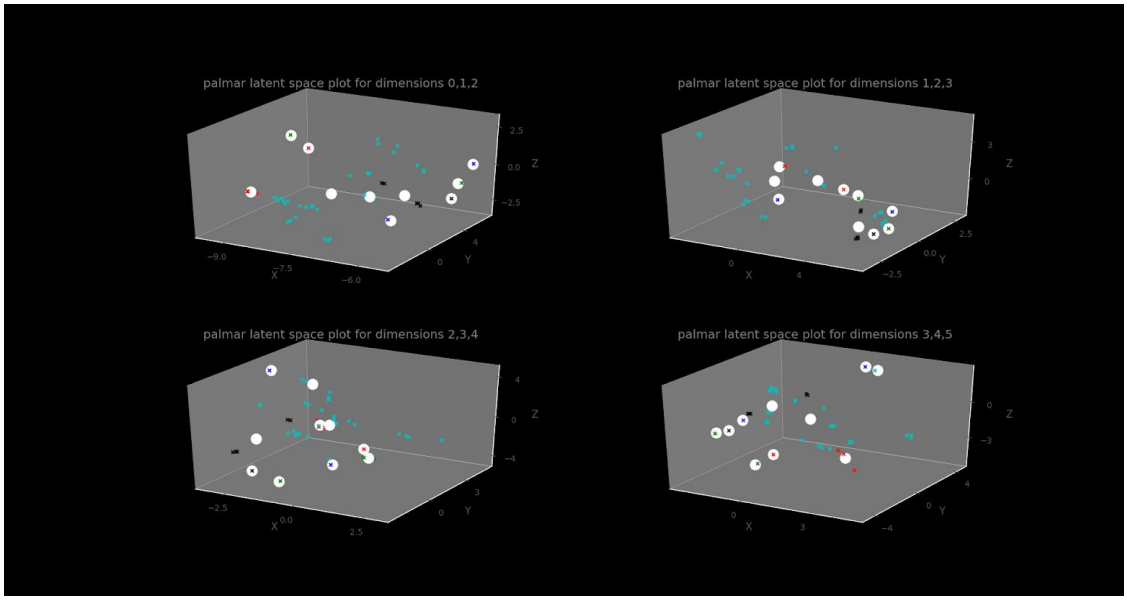

palmar latent space plot for dimensions 0,1,2
palmar latent space plot for dimensions 1,2,3
palmar latent space plot for dimensions 2,3,4
palmar latent space plot for dimensions 3,4,5

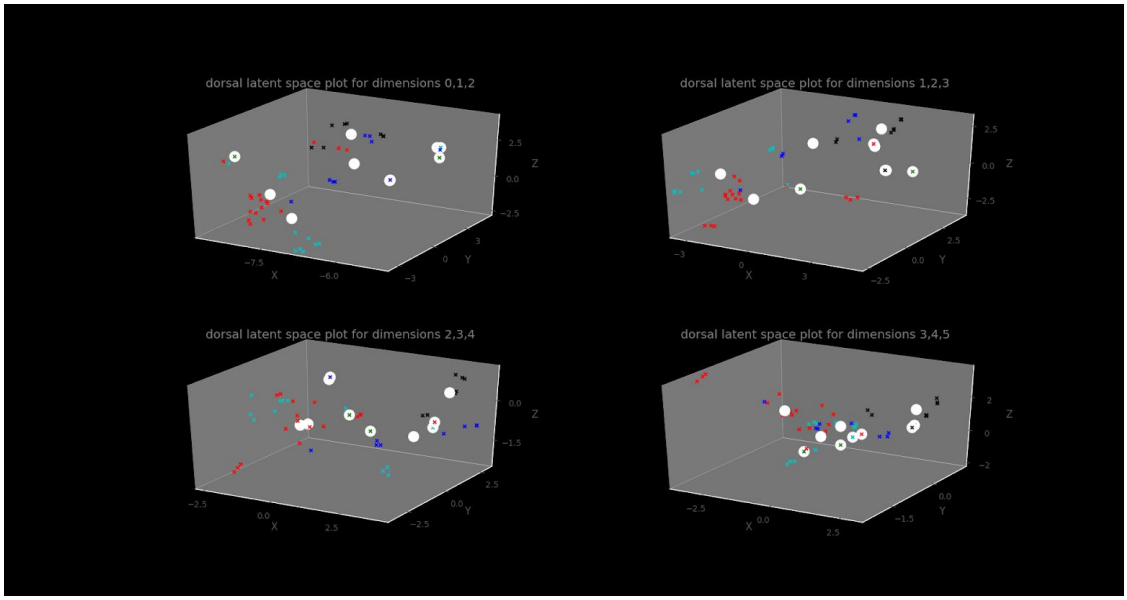
Query 3:


palmar latent space plot for dimensions 0,1,2
palmar latent space plot for dimensions 1,2,3
palmar latent space plot for dimensions 2,3,4
palmar latent space plot for dimensions 3,4,5


dorsal latent space plot for dimensions 0,1,2
dorsal latent space plot for dimensions 1,2,3
dorsal latent space plot for dimensions 2,3,4
dorsal latent space plot for dimensions 3,4,5

Query 4:





Task 3:

k	K	Images	Folder	Result Image
5	10	Hand_0008333.jpg, Hand_0006183.jpg, Hand_0000074.jpg	Labelled/Set 2	query-1.jpg
5	10	Hand_0003457.jpg, Hand_0000074.jpg, Hand_0005661.jpg	Labelled/Set 2	query-2.jpg

Query-1.jpg

Hand_0007739.jpg

Hand_0007738.jpg

Hand_0008333.jpg

Hand_0008176.jpg

Hand_0008334.jpg

Hand_0000074.jpg

Hand_0006183.jpg

Hand_0007737.jpg

Hand_0008335.jpg

Hand_0002991.jpg

Query-2.jpg

Hand_0005661.jpg

Hand_0003457.jpg

Hand_0000074.jpg

Hand_0007739.jpg

Hand_0007738.jpg

Hand_0005662.jpg

Hand_0003174.jpg

Hand_0008176.jpg

Hand_0003672.jpg

Hand_0003458.jpg

Task 4

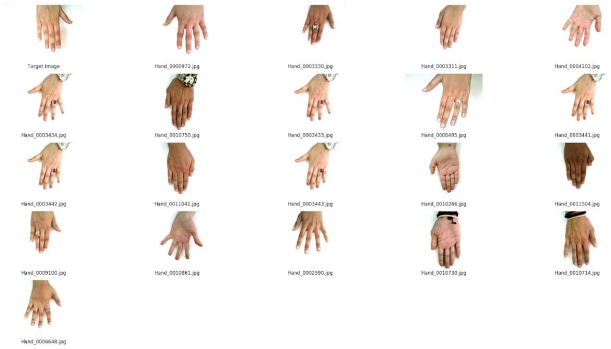
Classifier	Train Data	Test Data	Accuracy
SVM	Labelled/Set 2	Unlabelled/Set 2	51.0
Decision Tree	Labelled/Set 2	Unlabelled/Set 2	66.0
PPR	Labelled/Set 2	Unlabelled/Set 2	53.0

Task 5

1. **L = 10, k = 10, t : 20, Query Image: Hand_0000674.jpg**

2. **L: 10, k : 13, t : 20, Query Image: Hand_0000674.jpg**

3. **L: 5, k : 10, t : 20, Query Image: Hand_0000674.jpg**



Task 6

Query Image: Hand_0000674.jpg, **L**: 10, **k**: 10, t : 20

1. **Feedback** :SVM

   Before Feedback

   Relevant Images:

   Hand_0000675.jpg,Hand_0001375.jpg,Hand_0000609.jpg,Hand_0009217.jpg,Hand_0002545.jpg,Hand_0002541.jpg,Hand_0001051.jpg

   IRrelevant Images:

   Hand_0000724.jpg,Hand_0000723.jpg,Hand_0007027.jpg,Hand_0011215.jpg,Hand_0005157.jpg,Hand_0009581.jpg,Hand_0001195.jpg

After Feedback

2. **Feedback** : Decision Tree
   Before Feedback:

Relevant Images:
Hand_0001961.jpg,Hand_0003057.jpg,Hand_0003056.jpg,Hand_0003059.jpg,Hand_0003000.jpg,Hand_0003001.jpg,Hand_0008126.jpg
IRrelevant Images:

Hand_0011673.jpg,Hand_0003213.jpg,Hand_0003640.jpg,Hand_0001449.jpg,Hand_0001967.jpg,Hand_0010618.jpg,Hand_0006793.jpg

After Feedback

3. **Feedback**: PPR

Relevant Images:
Hand_0000675.jpg,Hand_0002060.jpg,Hand_0002853.jpg,Hand_0009741.jpg,Hand_0000007.jpg,Hand_0001184.jpg,Hand_0007448.jpg

IRrelevant Images:
Hand_0000049.jpg,Hand_0006806.jpg,Hand_0008695.jpg,Hand_0008581.jpg,Hand_0011396.jpg,Hand_0007761.jpg,Hand_0007996.jpg

After Feedback

4. **Feedback** : Probabilistic Relevance

Relevant Images:
Hand_0011315.jpg,Hand_0001954.jpg,Hand_0000759.jpg,Hand_0010306.jpg,Hand_0008126.jpg,Hand_0008434.jpg,Hand_0003799.jpg

IRrelevant Images:
Hand_0001751.jpg,Hand_0010290.jpg,Hand_0003764.jpg,Hand_0005151.jpg,Hand_0005152.jpg,Hand_0002448.jpg,Hand_0011158.jpg

After Feedback

Related work

In [1] the Local Binary Patterns of the 11k hands datasets are used as one of the features analysed by the SVM and the layers of Convolutional Neural Network. They propose the 11k hands dataset that contains information required for gender recognition and biometric identification. [2] proposes a method to extract such features that can be effective in matching different views of the same image. These features are invariant to scale and rotation in the image. [3] proposes Histogram of Oriented Gradients as features for the task of human detection. The method is based on calculation orientation of the gradients in the image and represent normalized histogram. [4] presents a probabilistic model that represents the probability associated with each topic which further is an abstraction of the underlying data that helps in easier computation of large datasets while preserving the information it conveys.

Conclusions

From this study, we've learned about various classifying techniques, and how effective they are based on the inputs provided. We've also learned about efficient index structures such as LSH that provide fast retrieval. Moreover, the choice of the features to use is highly dependent on what aspect of the image is discriminated against. In this, we choose color moments because we discriminate between dorsal and palmar images, which are maximally separated by color, and hence color moments is a good representation. However for general similarity search, HOG was found to be the better representation. Hence, it is understood that the choice of features is highly application specific, and there's no one-size-fits-all approach.

Bibliography

1. Afifi, Mahmoud. "11K Hands: gender recognition and biometric identification using a large dataset of hand images." *Multimedia Tools and Applications* 78.15 (2019): 20835-20854.

2. D. G. Lowe, "Distinctive image features from scale-invariant keypoints," International Journal of Computer Vision, vol. 60, no. 2, pp. 91–110, Nov 2004

3. N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 1, June 2005, pp. 886–893 vol. 1.

4. Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." *Journal of machine Learning research* 3, no. Jan (2003): 993-1022.

5. https://en.wikipedia.org/wiki/Color_moments

8. http://web.mit.edu/be.400/www/SVD/Singular_Value_Decomposition.htm

9.https://towardsdatascience.com/principal-component-analysis-for-dimensionality-reduction-115a3d157bad

10.https://towardsdatascience.com/understanding-locality-sensitive-hashing-49f6d1f6134

11. Brin, Sergey, and Lawrence Page. "The anatomy of a large-scale hypertextual web search engine." *Computer networks and ISDN systems* 30, no. 1-7 (1998): 107-117.

12. Lin, Frank, and William W. Cohen. "Semi-supervised classification of network data using very few labels." In *2010 International Conference on Advances in Social Networks Analysis and Mining*, pp. 192-199. IEEE, 2010.

Appendix

Task	**Members**
Infrastructure - Utilities, Shared code etc	All
Task 1	Tarun
Task 2	Dhruv
Task 3	Jay, Satyak
Task 4	Utkarsh, Harsh, Satyak, Jay
Task 5	Tarun, Dhruv
Task 6	All