CSCS 455– Data Mining and Data Warehousing

Course Project — Deadline: June 24th 11:59PM

Total Marks: 15

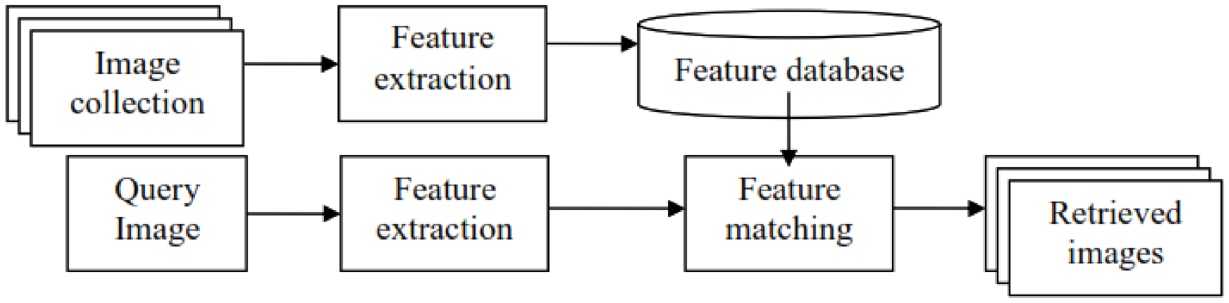
Instructions (Failure to follow instructions may result in mark deduction):

- The aim of this project is to give you hands-on regarding Python, feature engineering, and similarity/distance measures. The project is also intended to familiarize the students with the research report writing tool LaTex.
- **Download Data:** https://drive.google.com/file/d/18DgQtf0AYQHyRYwJhqtGC-R7Leohh5dA/view?usp=sharing
- Use WinRAR or any other software to extract all images. (If it returns a warning, ignore that.)
- *The dataset is collected from various internet sources so do not distribute it without permission from the original authors.*
- You will be implementing everything in Python. You can use Google Colab if you do not know how to setup a Python environment. Otherwise, install Anaconda Python Distribution on your local machine.
- Submit a report on Moodle in PDF format generated via LaTex, clearly mentioning question/ part number against your answers.
- For LaTex, you can use free online tool at https://www.overleaf.com/
- Once you create your account on overleaf, upload the attached zip template to start your report. All examples on how to include tables, references, figures, and code are given in the report to get you started. Delete anything that you do not need.

Problem Description

In this project, you are given a database (folder) of images. Your task is to develop an algorithm that can retrieve images similar to the query image. Unlike web-based search engines, your query is not a textual string. Instead, it is itself an image.

A naïve way of approaching this problem is to compare the query image with each of the image in the dataset using any appropriate distance measure (see Appendix A). Since image is a very high dimensional data, this approach is not feasible. Instead, a desirable mechanism is to project the image data in a low dimensional *"feature space"*. The retrieval is then done by comparing the features of the query image with the features of images in the dataset as shown below:



There are two modules of the system namely, (i) feature database creation (ii) retrieval. In these two modules, there are two essential components on which the performance of the system depend, which are image feature and the similarity measure used between the images.

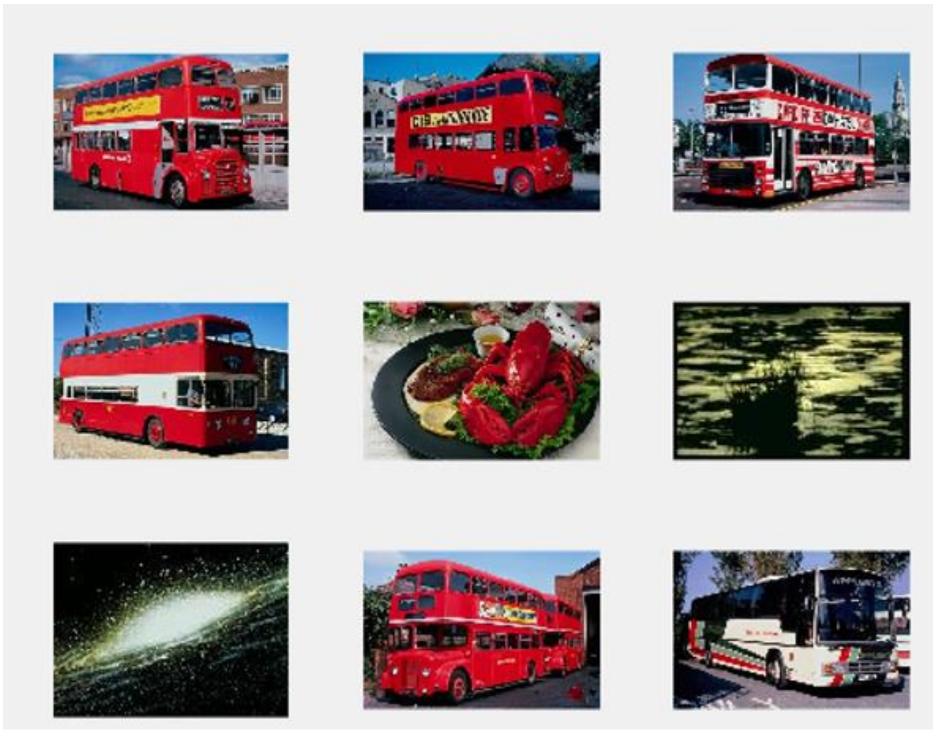
Feature vector: There can be a wide variety of the features that can be used to represent the image content in a low dimensional space. One such feature is image histogram. However, histogram does not contain any special information and two entirely different images can have exactly the same histogram (See below).

To introduce special information with histogram, you can divide the image into 9 small regions and compute the **histogram of each channel** of a region separately and make a single feature vector. Assuming you have an eight bit color image, this will result in a feature vector of length $2^8 \times 3 \times 9 = 6912$ per image. This feature vector of length 6912 is still a very large. You will also be working on a way to further reduce these feature dimensions (hint: PCA). See Appendix B for examples of more sophisticated region division strategies and feel free to implement any of them if you want.

Furthermore, image collection may contain images of varying sizes. Your solution should be such that it can correctly retrieve the same copy of an image even if it is stored at different resolutions.

Similarity measure: Select an appropriate similarity measure best suited for the problem. Compare results obtained by using at least three different similarity measures.

Interaction with the system: You will be writing a python script that takes two parameters. (1) a path of an image from ***"test"*** folder and (2) distance measure to use. Then it will return top 9 most similar images to the query from the database, in ascending order of the distance (i.e, most similar image would be the first image and so on), in a 3×3 grid (see below). This first image is always going to be query image itself as it also exists in the database.



What to submit: You will be submitting the PDF generate via LaTex, which will include all the codes for each task, results for each query image in a 3×3 grid, and your findings/discussion.

Appendix A: Distance and Similarity measures

Sum of Absolute Difference: is the distance between two points (p, q) in any dimension of space and is defined as:

$$d(p,q) = \sum_{k=1}^{n} |p_k - q_k|$$

Euclidian Distance: is the distance between two pints (p, q) in any dimension of space and is the most commonly used distance. When data is dense or continuous, this is the best proximity measure.

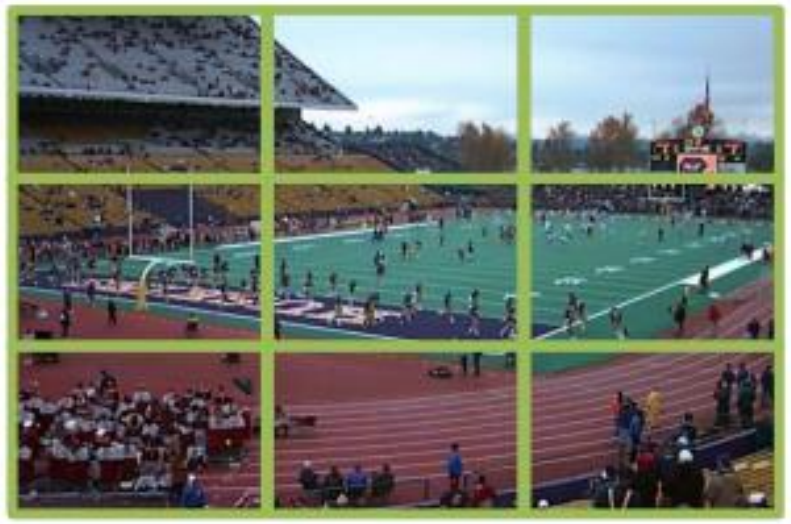
$$d(p,q) = \sqrt{\sum_{k=1}^{n} (p_k - q_k)^2}$$

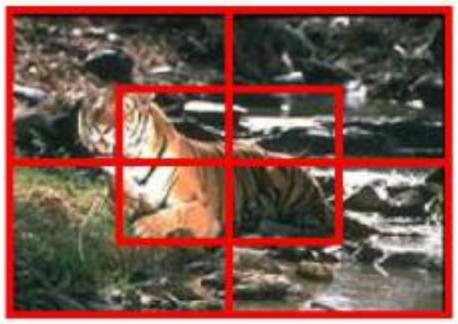
Cosine Similarity: is often used when comparing two documents against each other. It measures the angle between the two vectors. If the value is zero, the angle between the two vectors I 90 degrees and they share no terms. If the value is 1, the two vectors are the same except for magnitude. Cosine is used when data is sparse, asymmetric and there is a similarity of lacking characteristics. The cosine distance is $1 - consineSimilarity$.

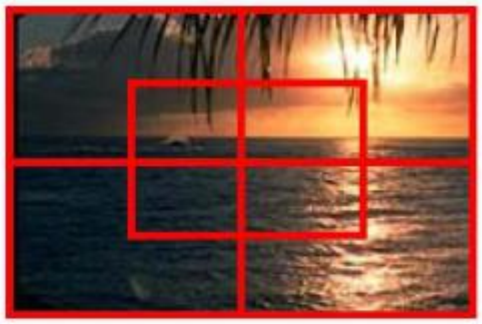
$$cos(p,q) = \frac{p.q}{||p||\,||q||}$$

Feel free to use any other distance measure.

Appendix B: Image Partition Strategies







Rectangles can overlap.

Task 1: (5 marks): Histogram Features

(Hint: You do not need to compute features for test image, given that it is part of images database. You can use this info to your advantage.)

- Write a function that will take an image as input, convert it into a grid of 3×3, and compute histogram for each channel. The function would return the final feature vector of histograms of each grid in the image. The size of the feature vector would be $2^8 \times 3 \times 9 = 6912$
- Write a function that takes a list of images names as input. For each image, the function is supposed to read it, convert it into a numpy array, and send the feature conversion function defined above. Each feature vector is then concatenated into a list and the final list is stored on hard drive (see how to pickle data in python). We will call this **feature database.** Also, save the image path in a separate list in a way that each image names' index matches to that of features list. If you have 1000 total images, you will have a 1000×6912 array of features and a list of length 1000 containing image paths.
- Write a function that takes a query image path and a distance measure to use as input, reads the query image and converts it to a feature vector of length 6912. Then for each feature vector in **feature database,** it computes the distance with query image features using the distance measure passed as parameter. Finally, it returns the top 9 most similar images.
- In your main function, take input from the user the test image path (query image) and a distance measure to use. Then call the function defined above which returns 9 most similar images. Show those images in a 3×3 grid and save it to hard drive. You have 3 test images and 3 distance measures to choose from. This way, you will have 9 results in total. Save the results on hard drive. Also, compute the time it took to match and retrieve the images using each distance function.
- Include all the code files in LaTex report. See LaTex template on how to include Python scripts in the report directly instead of copying and pasting the code.
- Include all 9 results (3 for each distance metric where each result is a 3×3 grid of 9 results) in the LaTex report. See the template on how to include an image.
- Make a table in LaTex report to mention the time it took to retrieve an image using all three distance metrics. (take any query image to report this time but use the same image for rest of the tasks as well to report this query time).
- Comment on the best and worst distance metric based on the results you have retrieved.

Task 2: (5 marks): Dimensionality Reduction with PCA

(Hint: You do not need to compute features and PCA for test image, given that it is part of images database. You can use this info to your advantage.)

- Write a function that will load the **feature database** stored as a result of Task 1 above. Then apply PCA and only select 1000 components. This way, for each image, the feature vector is no more of length 6912 and is reduced to 1000 dimensions. Save this **PCA reduced feature database** to hard drive. Also save the PCA object on the hard drive so that later we can use it to for query image.
- Write a function that takes a query image path and a distance measure to use as input, reads the query image and converts it to a feature vector of length 6912. Then it loads the PCA object saved above and transforms the query image into 1000 dimensional principal components. Then for each feature vector in **reduced feature database,** it computes the distance with query image features using the distance measure passed as parameter. Finally, it returns the top 9 most similar images.
- In your main function, take input from the user the test image path (query image) and a distance measure to use. Then call the function defined above which returns 9 most similar images. But this time, use **PCA reduced feature database.** Show those images in a 3×3 grid and save it to hard drive. You have 3 test images and 3 distance measures to choose from. This way, you will have 9 results in total. Save the results on hard drive.
- Include all the code files in LaTex report.
- Include all 9 results (3 for each distance metric where each result is a 3×3 grid of 9 results) in the LaTex report.
- Make another entry in the table in LaTex report to mention the time it took to retrieve an image using all three distance metrics. (report this time for the same image that you used in Task 1).
- Comment on the best and worst distance metric based on the results you have retrieved.

Task 3: (5 marks): Deep Learning based Features

In this task, we will use pretrained deep learning models to compute features from raw images. TensorFlow has a great collection of such pretrained models. See below link on how to use them. Use any model that you like but try to use latest one (from 2021 preferably) as those would be more efficient and faster.

Module: tf.keras.applications | TensorFlow Core v2.9.1

- Define a model using any pretrained models. As we do not need final classification layer, set "include_top = False". See tf.keras.applications.efficientnet_v2.EfficientNetV2B0 | TensorFlow Core v2.9.1 for an example.
- Write a function that will take an image as input, reads it, and passes it to the deep learning model defined above. The deep learning model will return a feature vector. The size of the feature vector would be different based on the model you chose.
- Write a function that takes a list of images names as input. For each image, the function is supposed to read it, convert it into a numpy array, and send it to the deep learning feature conversion function defined above. Each feature vector is then concatenated into a list and the final list is stored on hard drive. We will call this **deep feature database.** Also, save the image path in a separate list in a way that each image names' index matches to that of features list. If you have 1000 total images, you will have a $1000 \times n$ array of features and a list of length 1000 containing image paths. Here n is dependent on the deep learning model you chose.
- Write a function that takes a query image path and a distance measure to use as input, reads the query image and converts it to a feature vector of length 6912. Then for each feature vector in **deep feature database,** it computes the distance with query image features using the distance measure passed as parameter. Finally, it returns the top 9 most similar images.
- In your main function, take input from the user the test image path (query image) and a distance measure to use. Then call the function defined above which returns 9 most similar images. Show those images in a 3×3 grid and save it to hard drive. You have 3 test images and 3 distance measures to choose from. This way, you will have 9 results in total. Save the results on hard drive. Also, compute the time it took to match and retrieve the images using each distance function.
- Include all the code files in LaTex report.
- Include all 9 results (3 for each distance metric where each result is a 3×3 grid of 9 results) in the LaTex report.
- Make another entry in the table in LaTex report to mention the time it took to retrieve an image using all three distance metrics. (report this time for the same image that you used in Task 1 and 2).
- Comment on the best and worst distance metric based on the results you have retrieved.