**Esercitazione di approfondimento – BDM 2025**

Description: In this project, you will experiment with
• image features,
• vector models, and
• similarity/distance measures
• dimensionality curse
• clustering,
• indexing,
• classification

PROJECT TASKS:
• **Task 0**: In this phase of the project, we will use
– python as the programming environment,
– pytorch as the deep learning library,
– torchvision as the visual information extraction package, and
– you may also need numpy and scipy for array manipulation and other mathematical operations.

Familiarize yourself with these tools and languages. In particular, download and familiarize yourselves with the following pre-trained neural architecture and data set available through the torchvision package:

– pre-trained neural architecture: ResNet50 (with default weights)

– data set: BRAIN MRI; see the following URL for the description of the data set:

https://www.kaggle.com/datasets/orvile/brain-cancer-mri-dataset

Note that we will provide you a slighty modified version of the data set, where the data have been split into two parts. Please do not download the data from the URL. Instead use the version of the data we will provide to you.

In this phase, you are free to store the data however you wish: you can use a relational database (such as MySql), a no-SQL database (such as MongoDB), or create your own file/data structures.

• **Task 1**: Implement a program which, given an image filename and one of the following feature models, visualizes the image and then extracts and prints (in a human readable form) the corresponding feature descriptors:

– Color moments, CM10x10: Resize image to 300 x 100, partition the image into 10x10 grid, for each grid cell compute three color moments (mean, standard deviation, and skewness) for each channel in the RGB color space, and combine these color moments to obtain a unified 10 x10 x 3 x 3 = 900 dimensional feature descriptor.

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

– Histograms of oriented gradients, HOG: Map the image to gray scale, resize image to 300 x 10, partition the image into 10x10 grid, compute 9-bin (signed) magnitude-weighted gradient histogram (each bin corresponding to 40 degrees) for each grid cell, and combine these histograms into a 10 x 10 x 9 = 900 dimensional feature descriptor. You can use $\{-1, 0, 1\}$ and $\{-1, 0, 1\}^T$ masks to obtain dI/dx and dI/dy for each pixel position in the grid cell. Please see the following papers for additional information on HOG features:
∗ https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf
∗ https://ieeexplore.ieee.org/document/6523794
∗ https://filebox.ece.vt.edu/ jbhuang/teaching/ece5554-4554/fa16/lectures/Lecture 23 ObjectDetection2.pdf
∗ https://personalpages.surrey.ac.uk/j.collomosse/pubs/Hu-CVIU-2013.pdf
– ResNet-AvgPool-1024: Resize image to 224 x 24; attach a hook to the output of the "avgpool" layer of the ResNet pre-trained architecture to obtain 2048 dimensional vector, reduce the number of dimensions of the vector to 1024 by averaging two consecutive entries in the vector.
– ResNet-Layer3-1024: Resize image to 224 x 224; attach a hook to the output of "layer3" layer of the ResNet pre-trained architecture to obtain 1024 x 14 x 14 dimensional tensor, convert this tensor to a 1024 dimensional vector by averaging each 14 x 14 slice.
– ResNet-FC-1000: Resize image to 224 x 224; attach a hook to the output of "fc" layer of the ResNet pre-trained architecture to obtain a 1000 dimensional tensor.

• **Task 2**: Implement a program which extracts and stores feature descriptors for all images in the data set.

• **Task 3**: Implement a program which, given an image filename and a value "k", returns and visualizes the most similar k images based on each of the visual model -you will select the appropriate distance/similarity measure for each feature model. For each match, also list the corresponding distance/similarity score.

• **Task 4:**
– Task 4 a: Using pre-trained RESNET50 neural network model, map even numbered (labeled) images in the part 1 data set into 5 different feature spaces and store the resulting data vectors:
∗ Color moments, CM10x10
∗ Histograms of oriented gradients, HOG
∗ ResNet-AvgPool-1024
∗ ResNet-Layer3-1024
∗ ResNet-FC-1000
In the database, store not only the image filenames and feature vectors, but also the "dx type" labels.
– Task 4b: Implement a program which, given (a) a part 1 or part 2 image filename, (b) a user selected feature space, and (c) positive integer k, identifies and visualizes the most similar k part 1 images, along with their scores, under the selected feature space.

• **Task 5:**

– Task 5a: Implement a program which, given (a) a part 2 query image file, (b) a user selected feature space, and (c) positive integer k, identifies and lists k most likely matching labels, along with their scores, under the selected feature space.

• **Task 6** (LS1): Implement a program which (a) given one of the feature models, (b) a user specified value of k, (c) one of the three dimensionality reduction techniques (SVD, LDA, k-means) chosen by the user, reports the top-k latent semantics extracted under the selected feature space.
– Store the latent semantics in a properly named output file
– List imageID-weight pairs, ordered in decreasing order of weights

. • **Task 7**
• Task 7a: Implement a program which computes and prints the "inherent dimensionality" associated with the part 1 images.
• Task 7b: Implement a program which computes and prints the "inherent dimensionality" associated with each unique label of the part 1 images.

• **Task 8**: Implement a program which,
– for each unique label l, computes the corresponding k latent semantics (of your choice) associated with the part 1 images, and
– for the part 2 images, predicts the most likely labels using distances/similarities computed under the label-specific latent semantics.
The system should also output per-label precision, recall, and F1-score values as well as output an overall accuracy value.

• **Task 9**: Implement a program which,
– for each unique label l, computes the corresponding c most significant clusters associated with the part 1 images (using DBScan algorithm); the resulting clusters should be visualized both
∗ as differently colored point clouds in a 2-dimensional MDS space, and
∗ as groups of image thumbnails.

• **Task 10**: Implement a program which,
– given part 1 images,
∗ creates an m-NN classifer (for a user specified m),
∗ creates a decision-tree classifier,

For this task, you can use feature space of your choice.
– for the part 2 images, predicts the most likely labels using the user selected classifier.
The system should also output per-label precision, recall, and F1-score values as well as output an overall accuracy value.

• **Task 11;**
– 11a: Implement a Locality Sensitive Hashing (LSH) tool (for Euclidean distance) which takes as input (a) the number of layers, L, (b) the number of hashes per layer, h, and (c) a set of vectors as input and creates an in-memory index structure containing the given set of vectors. See
"Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions" (by Alexandr Andoni and Piotr Indyk). Communications of the ACM, vol. 51, no. 1, 2008, pp. 117-122.

– 11b: Implement a similar image search algorithm using this index structure storing the part 1 images and a visual model of your choice (the combined visual model must have at least 256 dimensions): for a given query image and integer t,
∗ visulizes the t most similar images,
∗ outputs the numbers of unique and overall number of images considered during the process.