Introduction to Computer Vision (ECSE 415)
Assignment 3: Classifiers, Object Recognition

DEADLINE: Feb 23, 2024
Please submit your assignment solutions electronically via the **myCourses** assignment dropbox.

The submission should include a single Jupyter notebook. More details on the format of the submission can be found below. Submissions that do not follow the format will be penalized 10%.

The assignment will be graded out of a total of **100 points**. There are *50 points* for accurate analysis and description, *40 points* for bug-free and clean code, and *10 points* concerning the appropriate structure in writing your report with citations and references.

Each assignment will be graded according to defined rubrics that will be visible to students. Check out MyCourses, the "Rubrics" option on the navigation bar. You can use **OpenCV**, **sklearn**, **skimage**, **Numpy**, and **Pytorch** library functions for all parts of the assignment except those stated otherwise. Students are expected to write their own code. (Academic integrity guidelines can be found **here**).

Assignments received late will be penalized by 10% per day.

Submission Instructions

1. Submit a single Jupyter notebook consisting of the solution of the entire assignment.
2. Comment your code appropriately.
3. Give references for all codes which are not written by you. (Ex. the code is taken from an online source or from tutorials)
4. Do not forget to run **Markdown** ('Text') cells.
5. Do not submit input/output images. Output images should be displayed in the Jupyter notebook itself.
6. Make sure that the submitted code is running without error. Add a **README** file if required.
7. If external libraries were used in your code please specify their name and version in the **README** file.
8. We are expecting you to make a path variable at the beginning of your codebase. This should point to your working local (or google drive) folder.
   Ex. If you are reading an image in the following format:

   ```
   img = cv2.imread ( '/content/drive/MyDrive/Assignment1/images/shapes.png' )
   ```

   Then you should convert it into the following:

   ```
   path = '/content/drive/MyDrive/Assignment1/images/'
   img = cv2.imread(path + 'shapes.png')
   ```

   Your path variable should be defined at the top of your Jupyter notebook. While grading, we are expecting that we just have to change the path variable once and it will allow us to run your solution smoothly. Specify your path variable in the **README** file.
9. Answers to reasoning questions should be comprehensive but concise.

1 CIFIAR10 Classification using SVM and Random Forest (50 points)

In this section, we will ask you to train Classifiers on the CIFIAR10 dataset. The CIFIAR10 dataset contains 60000 32x32 images. Ten classes are included, with 6000 images per class in it. Further information can be found on the dataset webpage https://www.cs.toronto.edu/ kriz/cifar.html. GPU is highly recommended for running code in this section.

1. Resize the train/test images to 64x64 and convert them to grayscale images. Compute HoG features with cells of 8x8 pixels, blocks of 4x4 cells, and 4 bins. This should generate a feature vector of size 1600 per image, which can be regarded as features for training classifiers.

2. Fit a non-linear SVM classifier with default hyperparameters on the features and the class features of the training images.

3. Predict labels of the test images by feeding the test features to the trained classifier and calculate classification accuracy.

4. Tune values of hyperparameters 'gamma' and 'C' to observe the accuracy change and select the hyperparameters with the highest test accuracy. Display your fine-tuning process by listing all the test cases with their parameter and corresponding accuracy.

5. Fit a Random Forest(RF) classifier (set n_estimators=10, max_depth=5, and criterion='entropy') on the features and the class labels of the training images.

6. Predict labels of the test images by feeding the test features to the trained classifier and calculate classification accuracy.

7. Compare the performance of SVM and RF. Experiment training both classifiers with a range of random states(different values for random_state). Evaluate the stability within the random state. List the strengths and weaknesses of each model.

2 Face Detection (50 points)

In this section, you will work on face detection. To achieve this purpose, you are required to compute EigenFaces and use the Viola-Jones detector to identify faces. We have provided a subset from CelebA(1) face dataset under the folder Q2 part1. The subset contains 100 color images.

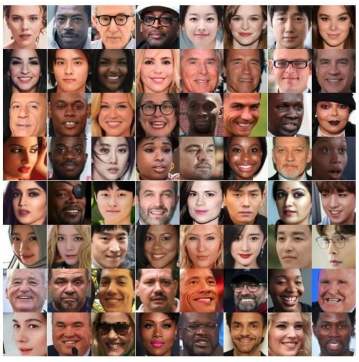

Figure 1: Sample images from CelebA dataset.

1. Read this subset into the code environment and convert all images into grayscale.
2. Implement the Snapshot method for PCA (covered in Lecture 8, Slide 55) from scratch using Numpy. Display the first five face images.

3. Use a sliding window method to detect faces in the image**??**, which is named Person.png under folder Q2 part2. Use the result from the previous step to compute the distance in the eignspace between the window contents and your training data.
4. Set a threshold to detect faces and select the best-performed value. Show your fine-tuned process.
5. Label the detected images with bound boxes and display the final result image with labels.
6. Use an existing implementation of the Viola-Jones face detector to detect faces on the same image. Compare the result with the method you implemented.
7. Evaluate your predicted result in both methods(True/False, Positive/Negative).
8. Explain under what conditions the Viola-Jones detector works when PCA does not.

Figure 2: Face Detection on Multi-person image.

9. Evaluate the performance of this model and explain the steps that this network took to achieve the final result.

References

[1] LIU, Z., LUO, P., WANG, X., AND TANG, X. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)* (December 2015).