Introduction to Computer Vision (ECSE 415)
Assignment 3: Face Detection and Classification

Due date: 11:59PM, March 15, 2021

This assignment will give you the opportunity to practice using eigenvectors to perform face detection and identification. You will work with a set of images containing faces, use principal components analysis (PCA) to represent them, then use that information to automatically detect those same faces in unseen data. You will then compare your implementation's performance to that of the Viola-Jones face detector (use publicly-available code; cite appropriately).

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%**. Attempt all parts of this assignment. The assignment will be graded out of total of **65 points**. Note that you can use any of the OpenCV and scikit-learn functions shown during tutorial sessions for this assignment, unless stated otherwise.

For this assignment, each student will work alone and is expected to write their own code. (Academic integrity guidelines can be found here).

Note: Assignments received up to 24 hours late will be penalized by 30%. Assignments received more than 24 hours late will not be graded.

Instructions for the Report

1. Submit one single jupyter notebook for the whole assignment. It should contain everything (code, output, and answers to reasoning questions) [1]
2. Write answers section-wise. The numbering of the answers should match the questions exactly.
3. Comment your code appropriately.

[1] If you are facing some memory and RAM issues, you are free to submit multiple notebooks as long as you clearly mention the reason for doing so.

4. The answers to the reasoning questions should be brief but comprehensive. Unnecessarily lengthy answers will be penalized. Use *markdown cell* type in jupyter notebook to write the answers.

5. If external libraries were used in your code please specify its name and version in the README file.

6. Run the whole jupyter notebook one last time using *Cell-> Run All*. This will run all cells in the jupyter notebook and show all outputs. We expect that when we run your jupyter notebook, we should be able to reproduce your results.

1 Data (5 Points)

For this assignment, you will have access to face images from a subset of the publicly available Color FERET Database [1] . You can download this dataset from google drive using this link This dataset is provided along with the assignment. Whereas the complete database contains images from approximately 1000 different subjects, the subset of the dataset that you are provided with contains images from only 52 subjects. The subset is arranged into several folders each containing images of a specific subject in different poses. The number of images per subject varies from 32 to 96. Each image is 768×512 pixels and the files are in PPM format. Convert images to gray-scale and down-sample images by the scale of 4. **(1 Point)**

You now need to prepare your dataset for a face recognition and detection system. Randomly separate the images into training and test sets as described below. A new random selection should be made by your program every time the system is retrained [2].

- **Train set:** For each subject, randomly select 80% of the total images given for that subject. This will be used as the training set. **(1 point)**
- **Test set:** All the remaining images from the given subset which are not used in the training set will be used as test images. **(1 point)**

Display total 10 random images. Plot histogram of the frequency of each image class (in this case the subject) distribution. **(2 Points)**

2 Eigenface Representation (25 Points)

You are now ready to create an eigenface representation for your training dataset through PCA. Please note that you are **not** allowed to use the in-built PCA

[2] For debugging purposes, you can fix your random seed. This should allow you get same split during different runs.

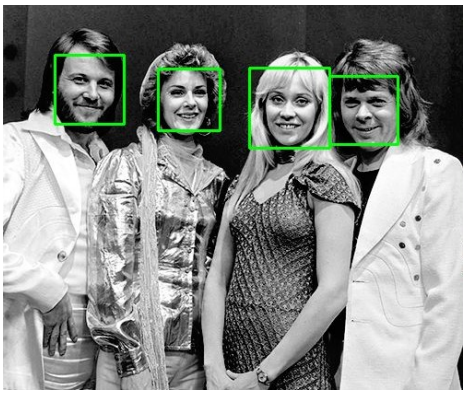

Figure 1: Face Detection: Example image with bounding boxes around the detected faces.

function in OpenCV/Scikit-Learn. You should implement the efficient Snapshot method for PCA (covered in class, Lecture 9, Slide 55) from scratch using numpy. **(15 points)**

Plot the fraction of total variance against the number of eigenvectors [3] **(2 points)**. Plot the normalized variance (eigenvalues) against the eigenvector index used for computation. **(2 points)**. Do you need all the vectors to represent the data? Discuss **(3 points)**. Display the first 5 eigenfaces **(5 points)**.

3 Classification (15 Points)

For every testing image, find the nearest neighbour (L2 distance), and check whether both images belong to the same person. To estimate the accuracy of this approach, determine what fraction of your test images has a neighbour that is actually of the same person? Compute the accuracy both in the original high dimensional pixel space and then in the eigenspace, and compare the accuracy values. Would you expect there to be a significant difference? **(10 points)**

You will now use a linear SVM classifier in the eigenspace. Use the training dataset to fit the classifier and the testing dataset to test the classifier. Compare the accuracy of this classifier with the nearest neighbour classifier used previously **(5 Points)**.

4 Face Detection (20 Points)

You will now detect all the faces in the given group image (Schitts_Creek_group.jpg) using PCA. Use a sliding window to detect the faces. Set a threshold on the dis-

[3] Refer to Tutorial 5 for more details.

tance in eigenspace between the window contents and your training data (Refer to slide 63 of Lecture 9). Try different values of thresholds and use the one which gives you good results. Display your image with bounding boxes around the detected faces for the best threshold. (ex. Figure 1) [4] **(15 points)**.

Use an existing implementation of the Viola-Jones face detector, and compare the results with your detector. Comparisons should be made in terms of the number of true positives, false positives and false negatives (See Appendix). Display the image with bounding boxes around the detected faces. Under what conditions would you expect the Viola-Jones detector to work when PCA does not? **(5 points)**.

Appendix

A bounding box is considered as a true positive if it contains a face image otherwise, it is regarded as a false positive. A missed face (no detected bounding box around) is considered a false negative.

[4] You can use any online available code for bounding box generation. Please cite the source for this in your report.