CA 2: Face Recognition with PCA and LDA crafted from Scratch for Feature Extraction followed by KNN, SVM, CNN and GMM for Recognition

In this assignment, you will work individually to construct a face recognition system using CMU PIE dataset and face photos taken by yourself. You should first apply Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) to perform data dimensionality reduction (Feature Extraction) and visualization. Then, you should apply K-Nearest-Neighbour (KNN), Support Vector Machine (SVM) and Convolutional Neural Networks (CNNs) for classifying the face images, and Gaussian Mixture Model (GMM) for grouping the face images.

Before you start the assignment, please read through the following guidelines so that you understand the requirements. Please follow the guidelines strictly to avoid unnecessary mark deductions:

1. Implement your codes with Python (recommended) or MATLAB (or any other languages you deem comfortable with).
2. For questions that require you to show your code, make sure your codes are **clean** and **easily readable** (add meaningful comments, comments are **markable**). Embed your code into your answer with screenshots.
3. Your submission should be one single PDF file with the necessary code and results collated in a single file. Name your PDF file strictly to **"MATRICULATION NUMBER_YOUR NAME_CA2.pdf"**. Incorrect submission format or naming will result in a **direct** 20 points (out of 100) deduction.
4. Do submit your project on Canvas before the deadline: **5:59 pm (SGT), 17 April 2025**. There will be **3** submissions attempts restricted for every student.
5. Policy on late submission: the deadline is a strict one, so please prepare and plan early and carefully. Any late submission will be deducted 10 points (out of 100) for every 24 hours.
6. This is an **individual** project, do **NOT** share your solutions with others, we have zero tolerance for **plagiarism**.
7. You are **NOT** allowed to use the direct functions for **Part 1** (e.g., sklearn.decomposition.PCA for PCA transformation or sklearn.discriminant_analysis.LinearDiscriminantAnalysis for LDA transformation), but other than these, you may use the basic operations (e.g., SVD, eigendecomposition, etc.) in any packages.
8. Wherever the randomness is needed, please use the last 2 numbers of your matriculation number as the seed value for randomization (e.g., if your matric. No. is A1234567A, the seed value = 67).

Part 1: Feature Extraction with PCA and LDA (50%)

1. **Dataset Preparation (5%)**

   Please download the CMU PIE dataset from the Canvas. There are in total 68 different subjects. Please randomly choose 25 out of them by "random.sample(range(1, 69), 25)" with a seed value to be the last 2 numbers of your matriculation number. For each chosen subject, use 70% of the provided images for training and use the remaining 30% for testing. Besides the provided CMU PIE images, please take 10 selfie photos for yourself, convert to gray-scale images, resize them into the same resolution as the CMU PIE images and split them into 7 for training and 3 for testing. Put the 7 selfie-photos into the training set and 3 into the test set (i.e., you will have in total 26 categories, 25 CMU PIE subjects + 1 for yourself). Please indicate your 26 categories in your report and show your 10 selfie photos after preprocessing (gray-scaling and resizing).

2. **PCA for feature extraction and visualization (20%)**

   a. The size of the image is 32x32, resulting in a 1024-dimensional vector for each image. Write a function to implement PCA from scratch that takes a data matrix "X" and number of PCs "p" to retain as the inputs, and return a matrix of *p* PC vectors and the corresponding PCA transformed data matrix. Show the code for the PCA function. (5%)

   b. Apply PCA using your crafted function to your training set with number of PCs to be 2 and 3 respectively. Visualize the corresponding projected training data in 2d and 3d plots. Use different colours for different classes in the plots. Highlight the projected points corresponding to your photos in the plots. Also visualize the corresponding 2 and 3 eigenfaces used for the dimensionality reduction. (10%)

   c. Apply PCA using your crafted function to your training set with number of PCs to be 80 and 200 respectively to prepare for the Part 2. Show the corresponding code. (5%)

3. **LDA for feature extraction and visualization (25%)**

   a. Write a function to implement LDA from scratch that takes a data matrix "X" and number of the LDA projections "p" to retain as the inputs, and return a matrix of *p* LDA projection vectors and the corresponding LDA transformed data matrix. Show the code for the LDA function. (10%)

   b. Apply LDA using your crafted function to your training set with number of LDA projection vectors to be 2 and 3 respectively. Visualize the corresponding projected training data in 2d and 3d plots. Use different colours for different classes in the plots. Highlight the projected points

corresponding to your photos in the plots. Also visualize the corresponding 2 and 3 fisherfaces used for the dimensionality reduction. (10%)

c. Apply LDA using your crafted function to your training set with number of LDA projection vectors to be 9 and 15 respectively to prepare for the Part 2. Show the corresponding code. (5%)

Part 2: Pattern Recognition (50%)

1. **KNN for classification (10%)**
   a. Apply KNN to PCA transformed testing dataset for p = 80 and 200 respectively with k=1. Report the corresponding classification accuracy on CMU PIE test images and your own test photos separately. Show the corresponding code for either p=80 or p=200. (5%)
   b. Apply KNN to LDA transformed testing dataset for p = 9 and 15 respectively with k=1. Report the corresponding classification accuracy on CMU PIE test images and your own test photos separately. Show the corresponding code for either p=9 or p=15. (5%)

2. **GMM for clustering (10%)**
   a. Use the raw vectorized training images to train a GMM model with 3 Gaussian components. Visualize the clustering results of the testing data from the trained GMM in a 2d plot with a highlight of points corresponding to your test photos. (3%)
   b. Use the PCA transformed training images to train a GMM model with 3 Gaussian components for p = 80 and 200 respectively. Visualize the clustering results of the testing data from the trained GMM in a 2d plot with a highlight of points corresponding to your test photos, for p = 80 and 200 respectively. Show the corresponding code for either p=80 or p=200. (7%)

3. **SVM for classification (15%)**
   a. Apply *linear* SVM (implements "one-vs-the-rest" multi-class strategy) to raw vectorized face images (each image is a 1024-dimensional vector). Try different values of the penalty parameter C in $\{1x10^{-2}, 1x10^{-1}, 1\}$. Report both training classification accuracy and testing classification accuracy with different C. (3%)
   b. Apply *linear* SVM to PCA transformed feature representations for p = 80 and 200 respectively. Try different values of the penalty parameter C in $\{1x10^{-2}, 1x10^{-1}, 1\}$. Report both training classification accuracy and testing classification accuracy with different C and different p. Show the corresponding code for only one of cases. (3%)

c. Apply *linear* SVM to LDA transformed feature representations for p = 9 and 15 respectively. Try different values of the penalty parameter *C* in {$1x10^{-2}$, $1x10^{-1}$, 1}. Report both training classification accuracy and testing classification accuracy with different *C* and different *p*. Show the corresponding code for only one of cases. (3%)

d. Discuss the effect of data dimension, feature learning method, and parameter *C* on the classification accuracy. (6%)

4. CNN for classification (15%)

Use the raw training face images (with a dimension of 32x32) to train a CNN with two convolutional layers, one fully connected layer, and one output layer. The first convolutional layer has 20 filters of size 5x5. The second convolutional layer has 50 filters of size 5x5x20. The first fully connected layer has 500 neurons followed by ReLU activation. The output layer has 26 neurons followed by softmax activation. Each convolutional layer is followed by a max pooling layer with a kernel size of 2x2 and stride of 2. The convolution operations are performed with a stride of 1 and no zero-padding. Report both training classification accuracy and testing classification accuracy. Show the detailed implementation.