Homework 3 **CSCE 633**
Due: 11.59pm on April 5, 2019

Instructions for homework submission
Please submit on eCampus a **single pdf** file containing your solutions.
a) Please typewrite in Latex the answers to the math problems. If this is not possible, please handwrite your solution *very clearly*, scan it and merge it to the final pdf file. Make sure that your solution is visible after scanning. Non-visible solutions will not be graded: we wouldn't like our TA to have to guess what you are writing :)
b) Please write a brief report for the experimental problems. At the end of the pdf file, please include your code. The code has to be directly converted instead of scanned (i.e. the text in the code must be selectable).
c) Please start early :)

Question 1: Convolution Operations

(a) Consider a 1-dimensional signal

$$F = [1 \quad 2 \quad 1 \quad 3 \quad 2 \quad 3 \quad 1 \quad 2 \quad 3 \quad 8 \quad 7 \quad 8 \quad 9 \quad 9 \quad 7 \quad 8]$$

The 1-dimensional convolution of the signal F with a filter F at point i is defined as:

$$(F \star W)[i] = \sum_{-\infty}^{\infty} F[j]W[i-j]$$

You can also find a nice visual representation in the link above:
`https://en.wikipedia.org/wiki/Convolution#Visual_explanation`

(a.i) (0.5 points) Calculate the convolution of F with filter $W_1 = [1 \quad 1 \quad 1]$, i.e., $F1 = F \star W_1$. What is the operation that filter W_1 performs? *Note:* You can ignore the edge elements and only compute the convolution for those elements which have all three neighbors.

(a.ii) (0.5 points) Calculate the convolution of F with filter $W_2 = [1 \quad 0 \quad -1]$, i.e., $F2 = F \star W_2$. What is the operation that filter W_2 performs? *Note:* You can ignore the edge elements and only compute the convolution for those elements which have all three neighbors.

(b) Consider an image $I \in \mathbb{R}^{5\times5}$ and a kernel filter $F \in \mathbb{R}^{3\times3}$. The convolution operation between I and F at pixel $[m, n]$ is defined as:

$$(I \star F)[m, n] = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} I[i, j]F[m-i, n-j]$$

(b.i) (2 points) *Implement* a 2-d convolution between the image and the filter. *Note:* You can ignore the edge pixels and only compute the convolution for those pixels which have all neighbors.

In the following, consider the following 5×5 image:

$$I = \begin{bmatrix} 164 & 188 & 164 & 161 & 195 \\ 178 & 201 & 197 & 150 & 137 \\ 174 & 168 & 181 & 190 & 184 \\ 131 & 179 & 176 & 185 & 198 \\ 92 & 185 & 179 & 133 & 167 \end{bmatrix}$$

(b.ii) (1 point) Using the function from **(b.i)**, compute the 2-d convolution between the above image I and the following filters :

$$F_1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

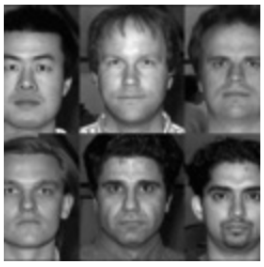
$$F_2 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

$$F_3 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

What do you observe in each case?

Question 2: Image processing for human faces

In this problem, we will process face images coming from the Yale Face Dataset, uploaded under the *Homework3_YaleDataset* folder in the Google Drive. This dataset contains images from 15 individuals. For each individual there are 11 images taken under a variety of conditions e.g., displaying a happy expression, wearing glasses etc.



(a) (2 points) *Implement* Principal Component Analysis (PCA) on the input images. Assume that the input vector of PCA contains all rows of an image stacked one on top of the other. You can use available libraries that calculate eigenvalues and eigenvectors of a matrix. *Hint:* Don't forget to normalize the data.

(b) (0.5 points) Plot a curve displaying the first k eigenvalues $\lambda_1, \ldots, \lambda_K$, i.e. the energy of the first K principal components. How many components do we need to capture 50% of the energy?

(c) (0.5 points) Plot the top 10 *eigenfaces*, i.e. the eigenvectors $\mathbf{u_k}$, $k = 1, \ldots, 10$ obtained by PCA.

(d) (Bonus) Select a couple of images from the data. Use the first k eigenfaces as a basis to reconstruct the images. Visualize the reconstructed images using 1, 10, 20, 30, 40, 50 components. How many components do we need to achieve a visually good result?
Hint: Reconstruction of an input vector $\mathbf{x}$ based on the eigenvectors $\mathbf{u_1}, \ldots, \mathbf{u_K}$ is given by the following expression $\mathbf{x} \approx \mathbf{x_0} + \sum_{k=1}^{K} c_k \mathbf{u_k}$, where $c_k = \mathbf{u_k^T x}$ is the projection of the input image to the k^{th} eigenvector.

(e) (1 point) *Perform face recognition:* Split the input data into training and testing *making sure that every person is included in each set.* Use any of the classification methods that we have learnt so far to recognize a person's identity. Use as input features the transformed feature space that resulted from PCA. Experiment with different number of PCA components through a 5-fold cross-validation. Report the recognition accuracy on the test set.

(f) (2 points) *Face recognition with CNNs:* Use a CNN to perform face recognition in the augmented Yale Face Dataset. Use the same split for the train and test set as in Question 2e. Experiment with different CNN parameters, e.g. # layers, filter size, stride size, activation function, dropout, pool size, etc.

(g) (Bonus) *Data augmentation:* Data augmentation is a way to increase the size of our dataset and reduce overfitting, especially when we use complicated models with many parameters to learn. Using any available toolbox or your own code, implement some of these techniques and augment the original Yale Face Dataset.
Hint: You can find more information in *hw3_DataAugmentationUsingDeepLearning.pdf* from *Homework 3* folder on Piazza and in the following link:
`https://machinelearningmastery.com/image-augmentation-deep-learning-keras/`