Introduction to Computer Vision (ECSE 415)
Assignment 4

Due: April 12^{th}, 11:59 PM

For this assignment you will in a group of 2. Please submit your assignment solutions electronically via the myCourses assignment dropbox. More details on the format of the submission can be found below. Attempt all parts of this assignment. The assignment will be graded out of total of **85 points**. Students are expected to write their own code. (Academic integrity guidelines can be found at `https://www.mcgill.ca/students/srr/academicrights/integrity`). Assignments received up to 24 hours late will be penalized by 30%. Assignments received more than 24 hours late will not be graded.

Submission Instructions

1. Submit two different jupyter notebooks (1 per person in a group) consisting of answers to 2 questions each from the assignment. All questions should be answered within these two notebooks. This will allow us to estimate the work done by each member of the group. Feel free to divide work as per your wish between members of the group. Submit one single zip file per group.
2. Comment your code appropriately.
3. Make sure that the submitted code is running without error. Add a README file if required.
4. If external libraries were used in your code, please specify their name and version in the README file.
5. Answers to reasoning questions should be comprehensive but concise.
6. Submissions that do not follow the format will be penalized 10%.

1 Segmentation (65 Points)

Use image 'flower.jpg' (Fig.1) for this question.

Figure 1: Image to be used for segmentation questions.

1.1 K-means clustering and Expectation Maximization

In this question, you will implement K-means and EM from scratch. **The algorithms should be implemented using only the numpy library.** You can use opencv and matplotlib libraries only to read and display images but not for clustering.

1.1.1 K-means clustering

- Implement K-means clustering from scratch. **(10 points)**
- Apply your implementation of K-means to the provided image with K=2 and K=3. Use R, G, B color channels of the image as three features. Display (a) the segmentation evolving during the first 5 iterations and (b) the final segmentation. **(5 points)**
- Convert the given image into gray-scale. Apply K-means to the provided image with K=2 and K=3. For every pixel, use gray-scale intensity as feature. Display (a) the segmentation evolving during the first 5 iterations and (b) the final segmentation. **(4 points)**
- Compare final segmentation maps of color and gray-scale images. Which feature results in better segmentation? **(1 points)**

1.1.2 Expectation Maximization – Gaussian Mixture Model

- Implement EM from scratch. **(15 points)**
- Apply your implementation of EM to the provided image with 2 and 3 Gaussian components. Use R, G, B color channels of the image as three features. Display (a) the segmentation evolving during the first 5 iterations and (b) the final segmentation. **(5 points)**

- Convert the given image into gray-scale. Apply EM to the provided image with 2 and 3 Gaussian components. For every pixel, use gray-scale intensity as feature. Display (a) the segmentation evolving during the first 5 iterations and (b) the final segmentation. **(4 points)**
- Compare final segmentation maps of color and gray-scale images. Which feature results in better segmentation? **(1 points)**

1.2 Normalized Graph-cut and Mean-shift Segmentation

You can use functions from OpenCV or skimage libraries for this question.

- Segment the given image using normalized graph-cuts. Vary the following parameters (try several values of each parameter): compactness and n segments (slic function), thresh (cut normalized function). Display segmentation results for several parameters and state their effect on the output **(10 Points)**.
- Segment the given image using mean-shift. Vary the following parameters (try several values of each parameter): ratio, kernel size, max dist. Display segmentation results for several parameters and state their effect on the output. **(10 points)**

2 Image Classification with Convolutional Neural Network (20 Points)

In this part, you will classify MNIST digits into 10 categories using a CNN. You may chose to run the code on GPU.

1. Use Pytorch class torchvision.datasets.MNIST to (down)load the dataset. Use batch size of 32. **(3 points)**
2. Implement a CNN with the layers mentioned below. **(6 points)**
    - A convolution layer with 32 kernels of size 3×3.
    - A ReLU activation.
    - A convolution layer with 64 kernels of size 3×3.
    - A ReLU activation.
    - A maxpool layer with kernels of size 2×2.
    - A convolution layer with 64 kernels of size 3×3.
    - A ReLU activation.
    - A convolution layer with 64 kernels of size 3×3.
    - A ReLU activation.

- A flattening layer. (This layer resizes 2D feature map to a feature vector. The length of this feature vector should be 4096.)
- A Linear layer with output size of 10.

(Suggestion: You can start with the code from the Tutorial and adapt it for the current problem.)

3. Create an instance of SGD optimizer with learning rate of 0.001. Use the default setting for rest of the hyperparameters. Create an instance of categorical cross entropy criterion. **(2 point)**

4. Train the CNN for 12 epochs. **(6 points)**

5. Predicts labels of the test images using the above trained CNN. Measure and display classification accuracy. **(3 points)**