Introduction to Computer Vision (ECSE 415)
Assignment 3: Advanced Computer Vision

DEADLINE: March 27, 11:59 PM

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 a total of **100 points**. There are *29 points* for accurate analysis and description, *61 points* for bug-free, clean code and correct implementation, 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**, **Scikit-Image**, and **Numpy** 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 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 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 Image Classification with Convolution Neural Network (CNN) and Naïve Bayes [53 points]

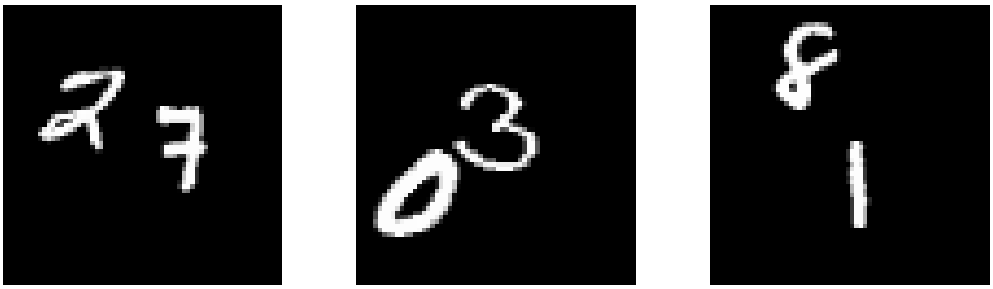
In this part, you will classify MNIST digits [1] into 10 categories using a CNN. You may chose to run the code on GPU.

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. [5 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.
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. [1 point]
4. Train the CNN for 10 epochs. Display loss and accuracy in each training step.[5 points]
5. Predict labels of the test images using the above trained CNN. Report classification accuracy. [3 points]
6. Show the effect of batch size on the classification accuracy of test images. Use a minimum of 4 different batch sizes (i.e. batch size = 4, 8, 12, 16). (*you will need to re-train the model*) [4 points]
7. Mention 2 other activation functions other than ReLU. Re-train the model using these activation functions instead of ReLU (throughout the network) and report the test classification accuracies. Discuss the results. [4 points]
8. Now, use the CNN developed above to recognize the digit in the test images (see Figure 1) - `'test1.png'`, `'test2.png'` and `'test3.png'`. Include the details of the important steps. (*You may need to pre-process the input image before using CNN to recognized the digit*) [4 points]

(a) test1.png (b) test2.png (c) test3.png

Figure 1: Three-handwritten test images

9. Now, Implement a probabilistic classifier that makes a Naïve Bayes assumption for the likelihood. Assume that you have a uniform prior. Now using this classifier, infer (i.e. mention the posterior probability for each class) on the handwritten test image in figure 1. (*You can use the numpy library to answer this sub-question (only for this sub-question))* [7 points]
10. Assume that the prior is the the frequency of occurrence of a class. At the test, show the effect of using class probability of the training sample (probability of a class is the frequency of occurrence of a class) as prior in Naïve Bayes on the same test data. Report the posterior class probabilities after using this prior. [4 points]
11. Using the trained CNN, detect (*i.e., you have to display two digits in each image*) the digits in 64×64 pixeled images (see figure 2) -`'detect1.png'`, `'detect2.png'` and `'detect3.png'`. Explain your results. [10 points]



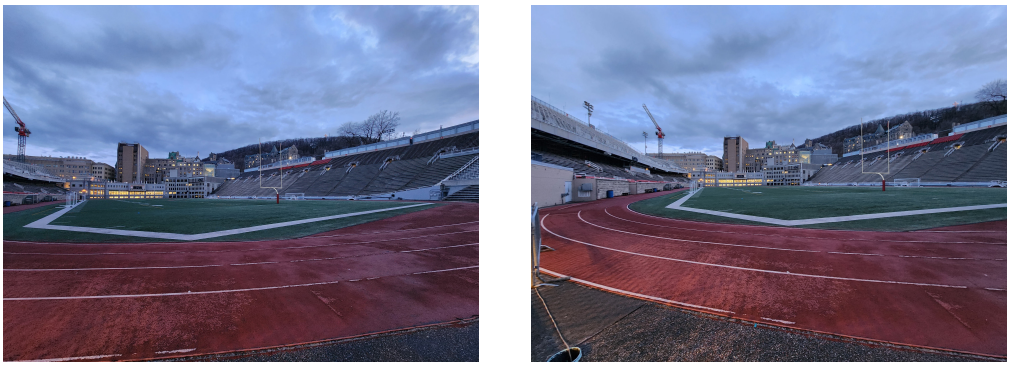
(a) detect1.png (b) detect2.png (c) detect3.png

Figure 2: Samples of 64×64 pixel images

12. Discuss the limitations of your CNN in the context where you CNN will fail to recognize digits. Show any one example. [3 points]

2 Stereo Vision - Epipolar Geometry [25 points]

Use a stereo image-pair shown in figure 3 for the following questions. You can use functions from OpenCV and matplotlib for this question.



(a) img1.jpeg (b) img2.jpeg

Figure 3: Stereo-image pair

1. Compute matching SIFT keypoints from a stereo image pair. [5 points]
2. Compute and display the epipolar lines for both images. [5 points]
3. Pick any one keypoint in the left image which has a correct match in the right image, and is on the corresponding epipolar line. Extract a patch of size (5×5) around this keypoint in the left image. [2 points]

4. Match the extracted patch to every 5×5 patch along the corresponding epipolar line in the right image. Use normalized cross-correlation metric for matching. [7 points]
5. Plot normalized cross-correlation values (on the y-axis) against an index of the patch in the left image (on the x-axis). Find the matching points with the maximum normalized cross-correlation value. Display found matching points in both images. [4 points]
6. Are the matching points you found the correct ones? Explain. [2 points]

3 Motion Algorithm [12 points]

Use the given two video frames 'frame1.png' and 'frame2.png' shown in Figure 4[*Source: Snapshots taken from YouTube Video*] for this question. You can use functions from OpenCV and matplotlib for this question.

(a) frame1.png (b) frame2.png

Figure 4: Video frames be used for testing motion algorithm.

Multi-resolution Lucas-Kanade optic flow detection

1. Extract good points to track from 'frame1.png' using Harris corner detection algorithm. Use openCV function `goodFeaturesToTrack` and set parameter value maxCorners=500. Search optimal values for the rest of the parameters. Let us call the detected set of points: p1. [2 points]
2. Compute the optical flow between 'frame1.png' and 'frame2.png' at the points in p1. Use the openCV function `calcOpticalFlowPyrLK`. Search for optimal values for the parameters. [3 points]
   Note that the function `calcOpticalFlowPyrLK` returns `nextPts` (a set of shifted positions of each point in p1 which we refer as p2), status (whether the search of shifted position is successful or not) and err (error measure between p1 and p2). Please read the manual for more details [1].)
3. Display the optical flow image. [1 point]
4. Vary the maximum pyramid level from 0 to 10 (in steps of 1) in the function `calcOpticalFlowPyrLK`. For each setting, compute the mean of the error at those points whose correspondence search is successful i.e. returned status is 1. Plot the mean (on y-axis) vs. pyramid level (on the x-axis). Discuss the trends you observe in the plot. [4 points]
5. Display the optical flow for each setting of the maximum pyramid level. Comment on the quality of the results. [2 points]

References

[1] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[1] `https://docs.opencv.org/2.4/modules/video/doc/motion_analysis_and_object_tracking.html#calcopticalflowpyrlk`