COMP/ENGN6528 Computer Vision - 2022
Computer-Lab 2(C-Lab2)

COMP/ENGN6528, 2022

Objectives:

This is CLab-2 for COMP/ENGN6528 Computer Vision. This Lab focuses on features, developing mid-level computer vision features, and using Deep Learning to learn features to perform a classification task.

For this the second part of this lab we highly recommend that you use PyTorch. Please discuss with your tutor if you plan to use anything else.

Special Notes:

1. Each computer lab has three weeks for submission, but only two weeks of classes: session-A and session-B. Tutors/Lab instructor will provide basic supervision to both sessions. Please attend both sessions to get support. Piazza support is available throughout.
2. Your Lab will be marked based on the overall quality of your Lab Report (PDF). The report is to be uploaded to Wattle site before the due time, which is usually on Sunday 11:59pm of Week-3 session of your lab.
3. Your submission includes the lab report in PDF format as well as the Lab code that generating the experimental result.
4. It is normal if you cannot finish all the tasks within two 2-hour sessions — these tasks are designed so that you will have to spend about 9 hours to finish all tasks including finishing your Lab report. This suggests that, before attending the third lab session (in Week-2 of each CLab), you must make sure that you have almost complete 80%. It may take longer if you are not familiar with programming in Python and with object oriented concepts, or less time if you are already familiar with pytorch.

Academic Integrity:

You are expected to comply with the University Policy on Academic Integrity and Plagiarism. You are allowed to talk with / work with other students on lab and project assignments. You can share ideas but not code, you should submit your own work. Your course instructors reserve the right to determine an appropriate penalty based on the violation of academic dishonesty that occurs. Violations of the university policy can result in severe penalties.

Listing 1: harris.m

```
1  %%
2  % CLAB2 Task-1: Harris Corner Detector
3  % Your name (Your uniID)
4  %
5  sigma = 2; thresh = 0.01; % Parameters, add more if needed
6  % Derivative masks
7  dx = [-1 0 1;-1 0 1; -1 0 1];
8  dy = dx'; % dx is the transpose matrix of dy
9  % compute x and y derivatives of image
10 Ix = conv2(bw,dx,'same');
11 Iy = conv2(bw,dy,'same');
12
13 g = fspecial('gaussian',max(1,fix(3*sigma)*2+1),sigma);
14 Ix2 = conv2(Ix.^2,g,'same'); % x and x
15 Iy2 = conv2(Iy.^2,g,'same'); % y and y
16 Ixy = conv2(Ix.*Iy,g,'same'); % x and y
17
18 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
19 % Task: Compute the Harris Cornerness                              %
20 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
21
22 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
23 % Task: Perform non-maximum suppression and                       %
24 %       thresholding, return the N corner points                  %
25 %       as an Nx2 matrix of x and y coordinates                   %
26 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure 1: For Matlab Users.

CLab-2 Tasks

Task 1 Harris Corner Detector. (5 marks)

For Matlab Users:

1. Read and understand the corner detection code 'harris.m' in Fig 1.
2. Complete the missing parts, rewrite 'harris.m' into a Matlab function, and design appropriate function signature (1 mark).
3. Please provide comments on line #13 and every line of your solution after line #20 (0.5 mark). Specifically, you need to provide short comments on your code, which should make your code readable.
4. Test this function on the provided four test images (Harris-[1,2,3,4].jpg, they can be downloaded from Wattle). Display your results by marking the detected corners on the input images (using circles or crosses, etc) (0.5 mark for each image, 2 marks in total).

   Please make sure that your code can be run successfully on a local machine and generate results. If your submitted code cannot replicate your results, you may need to explain and demonstrate the results in person to tutors.
5. Compare your results with that from Matlab's built-in function corner() (0.5 mark), and discuss the factors that affect the performance of Harris corner detection (1 mark).

   In your Lab Report, you need to list your complete source code with detailed comments and show corner detection results and their comparisons for each of the test images.

For Python users:

1. Read and understand the above corner detection code (Fig. 2).

2. Complete the missing parts, rewrite them to 'harris.py' as a python script, and design appropriate function signature (1 mark).

3. Comment on block #5 (corresponding to line #13 in "harris.m") and every line of your solution in block #7 (0.5 mark). Specifically, you need to provide short comments on your code, which should make your code readable. Test this function on the provided four test images (Harris-[1,2,3,4].jpg, they can be downloaded from Wattle). Display your results by marking the detected corners on the input images (using circles or crosses, etc) (0.5 mark for each image, 2 marks in total).

   Please make sure that your code can be run successfully on a local machine and generate results. If your submitted code cannot replicate your results, you may need to explain and demonstrate the results in person to tutors.

4. Compare your results with that from python's built-in function cv2.cornerHarris() (0.5 mark), and discuss the factors that affect the performance of Harris corner detection (1 mark).

   In your Lab Report, you need to list your complete source code with detailed comments and show corner detection results and their comparisons for each of the test images.

Task 2 - Deep Learning Classification (10 Marks)

In this lab, we will train a CNN with the Kuzushiji-MNIST dataset using the PyTorch[1] deep learning framework. The Kuzushiji-MNIST dataset contains 70000 images: 59000 training images, 1000 validation images and 10000 testing images[2]. Images are 28×28 Greyscale.

Complete the following exercises:

1. Download the Kuzushiji-MNIST dataset from google drive: link

2. After loading the data using numpy, normalize the data to the range between (-1, 1). Also perform the following data augmentation when training:

   - randomly flip the image left and right.
   - zero-pad 4 pixels on each side of the input image and randomly crop 28x28 as input to the network.

3. Build a CNN with the following architecture:

   - 5×5 Convolutional Layer with 32 filters, stride 1 and padding 2.
   - ReLU Activation Layer.
   - 2×2 Max Pooling Layer with a stride of 2.
   - 3×3 Convolutional Layer with 64 filters, stride 1 and padding 1.
   - ReLU Activation Layer.
   - 2×2 Max Pooling Layer with a stride of 2.
   - Fully-connected layer with 1024 output units.
   - ReLU Activation Layer.
   - Fully-connected layer with 10 output units.

4. Set up cross-entropy loss.

5. Set up Adam optimizer, with 1e-3 learning rate and betas=(0.9, 0.999). (3 marks to here)

6. Train your model. Draw the following plots:

   - Training loss vs. epochs.

[1] Let tutor know if you want to use tensorflow which is acceptable.

[2] The reference for the dataset is: Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, David Ha, "Deep Learning for Classical Japanese Literature", arXiv:1812.01718.

- Training accuracy vs. epochs.
- Validation loss vs. epochs.
- Validation accuracy vs. epochs.

You can either use Tensorboard to draw the plots or you can save the data (e.g. in a dictionary) then use Matplotlib to plot the curve. (2 marks)

7. Train a good model. Marks will be awarded for high performance and high efficiency in training time and parameters (there may be a trade-off), good design, and your discussion. You are not allowed to use a pre-trained model, you should train the model yourself. You need to describe what exactly you did to the base model to improve your results in your report, and your motivation for your approach (no more that 1 page of text). Please include plots as above for training and validation loss and accuracy vs. epochs, as well as the final accuracy on the test set. Please submit the code and your trained model for this. Your performance will be verified. Please ensure you follow the test/train/validation splits as provided. You also must show your training and validation accuracy vs. epochs for this model in your report. Note that you may be asked to run training of your model to demonstrate its training and performance. (4 marks)

8. The main dataset site on github (https://github.com/rois-codh/kmnist) includes a series of results for other network models (under Benchmark & Results). How does your model compare? Explain why the ResNet18 model may produce better results than yours (or the other way around if this is the case). (1 mark).

In your Lab Report, you need to list your complete source code with detailed comments.

Resources:

- PyTorch training a classifier tutorial: https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html (Note this uses CIFAR-10, but you need to train Kuzushiji-MNIST.)
- PyTorch Documentation: https://pytorch.org/docs/stable/index.html
- Deeper models implementation: https://github.com/pytorch/vision/tree/master/torchvision/models
- Tensorboard tutorial: https://www.tensorflow.org/tensorboard/get_started

1 Notes

1. The lab report will be due on Sunday, 1 May 2022, 11:59pm. You are required to complete all tasks and answer all questions. Please submit a single zip file document and attach all your code. You are required to submit code including your best performing trained model, and Latex is recommended for the report (but not required). Name your pdf as Lab_2_uxxxxxxx.zip, replacing uxxxxxxx with your uni-ID.

2. This Lab is worth 15% of the total course assessment.

In [1]:
```
"""
CLAB2 Task-1: Harris Corner Detector
Your name (Your uniID):
"""
import numpy as np
```

In [2]:
```
def conv2(img, conv_filter):
    # flip the filter
    f_siz_1, f_size_2 = conv_filter.shape
    conv_filter = conv_filter[range(f_siz_1 - 1, -1, -1), :][:, range(f_siz_1 - 1, -1, -1)]
    pad = (conv_filter.shape[0] - 1) // 2
    result = np.zeros((img.shape))
    img = np.pad(img, ((pad, pad), (pad, pad)), 'constant', constant_values=(0, 0))
    filter_size = conv_filter.shape[0]
    for r in np.arange(img.shape[0] - filter_size + 1):
        for c in np.arange(img.shape[1] - filter_size + 1):
            curr_region = img[r:r + filter_size, c:c + filter_size]
            curr_result = curr_region * conv_filter
            conv_sum = np.sum(curr_result)  # Summing the result of multiplication.
            result[r, c] = conv_sum  # Saving the summation in the convolution layer feature map.

    return result
```

In [3]:
```
def fspecial(shape=(3, 3), sigma=0.5):
    m, n = [(ss - 1.) / 2. for ss in shape]
    y, x = np.ogrid[-m:m + 1, -n:n + 1]
    h = np.exp(-(x * x + y * y) / (2. * sigma * sigma))
    h[h < np.finfo(h.dtype).eps * h.max()] = 0
    sumh = h.sum()
    if sumh != 0:
        h /= sumh
    return h
```

In [4]:
```
# Parameters, add more if needed
sigma = 2
thresh = 0.01

# Derivative masks
dx = np.array([[1, 0, -1], [1, 0, -1], [1, 0, -1]])
dy = dx.transpose()

# computer x and y derivatives of image
Ix = conv2(bw, dx)
Iy = conv2(bw, dy)
```

In [5]:
```
g = fspecial((max(1, np.floor(3 * sigma) * 2 + 1), max(1, np.floor(3 * sigma) * 2 + 1)), sigma)
```

In [6]:
```
Iy2 = conv2(np.power(Iy, 2), g)
Ix2 = conv2(np.power(Ix, 2), g)
Ixy = conv2(Ix * Iy, g)
```

In [7]:
```
########################################################################
# Task: Compute the Harris Cornerness
########################################################################


########################################################################
# Task: Perform non-maximum suppression and
#       thresholding, return the N corner points
#       as an Nx2 matrix of x and y coordinates
########################################################################
```

Figure 2: For Pyhon Users.