CSE47201 Computer Vision Programming Assignment 1

Deadline : until 2022 Oct. 28, 23:59

- If you have any questions, please post it in the blackboard/discussion/PA1 Q&A board.
- Fonts in pink are recently added to prevent confusions in problems (Last update: 2022.09.28. 21:49).

1. A mafia sends you the mail containing an image (640x480 size) every Monday. The image contains a digit (0-9) in the **red** channel. The font size, type, color, etc is always the same; but the location of the digit changes every time (but the digit is always located around the center of the image). For example, you can download images in the link below:
   https://drive.google.com/file/d/1PH20U_lhM7A5js_2j7L7MMgmwVy5ASvZ/view?usp=sharing
   [20 points] Please clearly visualize digits contained in the five images downloaded from the link and manually comment what are the digits. (Please implement a python code for visualizing digits. Also, please manually add a comment for what are the digits).

2. Using the ``Bag-of-words'' model, please implement an object classifier that can classify 20 objects defined in the dataset: https://drive.google.com/file/d/1FmnFOJBN5ihcKC_WW7T99bN-reTrVu0-/view?usp=sharing

   You may need to use the multi-class SVM function by calling the function below:

   ```
   svm = cv2.ml.SVM_create()
   svm.train(hists, cv2.ml.ROW_SAMPLE, np.array(train_labels))
   svm.save(svm_model_file)
   ```

   - Please use the training database (in train folder) when training your model while using the testing database (in test folder) when evaluating your model.

   - **[20 points]** Please implement the object classifier and measure their accuracy. The accuracy measure is defined as follows:

     $$Accuracy = \frac{(\#\ of\ samples\ which\ are\ correctly\ predicted)}{(\#\ of\ total\ samples)}$$

     Please calculate the 'accuracy' of your 20 object classifier using the 'SVM' on the downloaded testing data.

   - **[15 points]** Please find the relationship between the 'number of codewords' and the overall 'accuracy' of your method. (You may need to draw graphs by changing # of codewords and calculating corresponding accuracy.)

- **[15 points]** Please use a 'histogram intersection kernel' for 'SVM' and measure its accuracy samely as above. (In above multi-class SVM code, the default kernel was the 'RBF kernel'). You can change the kernel by using the code below:

```
svm.setKernel(cv2.ml.SVM_INTER)
```

3. There is a random function $f: x \rightarrow y$ that maps 100-dimensional vector x towards a scalar value $y \in [0, 1]$. The door locker is implemented based on this function f, and you need to input the proper vector x to open the locked door. It will be opened if the output of the function $y = f(x) > 0.9$.

   We decided to generate a door hacker that can open any door locked based on the function f. The door hacker will operate based on the function $g: z \rightarrow x$ that maps 100-dimensional vector z towards 100-dimensional vector x. This vector x will be used as the input to the door locker. There is no restriction for the z vector other than the fact that its dimension is 100. Further, the door hacker can access the output of the door locker (ie. y) and can measure how the output would be changed by the input vector z (ie. $\frac{dy}{dz}$).

   [20 points] Please implement the code for opening the door within 50 trials for a function f with random weights. (Please use the DoorLock, DoorHack class below for implementing your solution.)
   [Notes] Please do not modify the DoorLock class; while you are allowed to modify the DoorHack class and any other codes. Please also assume that the DoorHack class cannot access the inside of the DoorLock class (This is the hacking scenario).
   [Hints] Please try to define the loss L (for example $L = (f(g(z)) - 1)^2$) and exploit the $\frac{dL}{dz}$ to update the z within 50 trials.

```
import torch
import torch.nn as nn

class DoorLock(nn.Module):
  def __init__(self):
    super().__init__()
    self.f = nn.Sequential(
        nn.Linear(100, 1),
        nn.Sigmoid()
    )
    for p in self.f.parameters():
      p.requires_grad = False

  def forward(self, x):
    y = self.f(x)
    if(y > 0.9):
      print('Opened!')
    return y

class DoorHack(nn.Module):
  def __init__(self, locker):
    super().__init__()
    self.g = nn.Sequential(
        nn.Linear(100, 100),
    )
    self.locker = locker

  def forward(self, z):
    y = self.locker(self.g(z))
    return y

num_trials = 50
locker = DoorLock()
hacker = DoorHack(locker)
```

4. **[10 points]** Please note and apply the below details.

   a. Send all files(.zip) via **Blackboard/Assignment/PA1** menu.

   b. Zip all files to **20220927_SeungryulBaek_cv_ass1.zip** and it needs to take each problem's .ipynb and result of .pdf files(note that three .ipynb and three .pdf files) and dictionary.npy, svmmodel.xml, nearest_neighbor.xml files.

      <Example>

      ```
      20220927_SeungryulBaek_cv_ass1.zip
      ├-- Problem1.ipynb
      └--Problem1.pdf
      └--    …
      └--dictionary.npy
      ├--nearest_neighbor.xml (You can omit this, if you didn't use K-NN)
      └--svmmodel.xml
      ```

      * IPYNB->PDF Tool: https://htmtopdf.herokuapp.com/ipynbviewer/

   c. Code should have some **comments** to increase the readability.