Programming Assignment-II

Computer Vision

Coding Standard and General Requirements

Code for all programming assignments should be **well documented**. A working program with no comments will receive **only partial credit**. Documentation entails writing a description of each function/method, class/structure, as well as comments throughout the code to explain the program flow. Preferred programming language for the assignment is **Python** with PyTorch framework for deep learning. You are free to use any other programming language and framework as well.

Submit by **15th of November 2022**, 11.59pm.

NOTE: This assignment is optional and can be used to earn BONUS points. There will be NO extension for this assignment.

Question 1: Autoencoder [2.5 pts]

Implement autoencoder using MNIST dataset. The input size of the images will be 28x28 with single channel. You will implement two different variations, one with fully connected layers (standard neural network), and the other with convolutional neural network.

Your tasks:

- Implement an autoencoder using fully connected layers. The encoder will have 2 layers (with 256, and 128 neurons) and the decoder will also have two layers (with 256 and 784 neurons). Train this network using MSE loss for 10 epochs. Compare the number of parameters in the encoder and the decoder. Show 20 sample reconstructed images from testing data in the report (2 image for each class) along with the original images.
- Implement a convolutional autoencoder for MNIST dataset. The encoder will have two concolutional layers, and two max-pooling layers followed by each convolutional layers. Use kernel size 3x3, relu activation, and padding of 1 to preserve the shape of the input feature map. The decoder will have three convolutional layers with kernel shape 3x3 and padding of 1 to preserve the feature map shape. The first two convolution layer will be followed by an upsampling layer, which will double the resolution of feature maps using linear interpolation. Train this network for 10 epochs. Compare the number of parameters in the encoder and the decoder. Also, compare the total parameters in this autoencoder with the autoencoder in the previous task. Show 20 sample reconstructed images from testing data in the report (2 image for each class) along with the original images. Also compate the reconstructed results with the previous autoencoder.

NOTE: You are free to choose any optimizer, but use the same optimizer for both the variations. Feel free to use the code shared in the first assignment for data loader and other base classes.

What to submit:

- Code

- A short write-up about your implementation with results (as indicated for each variation) and your observations from each training.

Question 2: Nearest Neighbor Classification [2.5 pts]

In this question, the task is to implement nearest neighbor classifier for digit classification. You will use the digit dataset available from sklearn library. There are around 1800 images in total with 10 digit classes, and each image is 8x8 sized with single channel. You will have to split the dataset into training and testing, keep 500 images for testing (you will have to choose them randomly with 50 images per class).

NOTE: We have not covered this topic in class yet, it will be discussed next week.

Sample code to load the dataset from sklearn,
from sklearn.datasets import load_digits
digits = load_digits()

Your tasks:

- Implement a nearest neighbor classifier using pixels as features. Test the method for classification accuray.
- Implement a k-nearest neighbor classifier using pixels as features. Test the method for k=3, 5, and 7 and compute classification accuracy.

NOTE: You can use L2-norm for distance between two samples.

What to submit:

- Code
- A short write-up about your implementation with results: 1) Accuracy scores for all the variations, 2) Compare all the variations using accuracy scores. Comment of how the accuracy changes when you increase the value of k.