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. Programming language for the assignment is **Python**. You can use standard python built-in IDLE, or CANOPY for the working environment. Other commonly used IDLEs are the following: PyCharm Community Edition, PyScripter, CodeSculptor, Eric Python, Eclipse plus PyDev.

Following libraries will be used extensively throughout the course:

- PIL (The Python Imaging Library), Matplotlib, NumPy, SciPy, OpenCV, python-graph.

For example, if you are asked to implement "Gaussian Filtering", you are not allowed to use a Gaussian function from a known library, you need to implement it from scratch.

We recommend PyTorch for this assignment.

Submit by **23rd of October 2023**, 11.59pm.

Question: Convolutional Neural Network (CNN) for Classification [5 pts]

Implement ConvNET using **PyTorch** for digit classification. Sample code files (two files) are given in the attachment. Fill the parts indicated clearly in the code. Output should be saved as **output.txt**. When you are asked to include convolutional layer, do not forget to include max pooling or average pooling layer(s) as well. If you want to use any other framework, you are free to do that. Remember, no base code will be provided for any other framework.

- STEP 1: Create a fully connected (FC) hidden layer (with 100 neurons) with Sigmoid activation function. Train it with SGD with a learning rate of 0.1 (a total of 60 epoch), a mini-batch size of 10, and no regularization.
- STEP 2: Now insert two convolutional layers to the network built in STEP 1 (and put pooling layer and Sigmoid activation too for each convolutional layer). Pool over 2x2 regions, 40 kernels, stride =1, with kernel size of 5x5. Train with same setting as STEP 1.
- STEP 3: For the network depicted in STEP 2, replace Sigmoid with ReLU, and train the model with new learning rate (=0.03). Re-train the system with this setting.
- STEP 4: Add another fully connected (FC) layer now (with 100 neurons) to the network built in STEP 3. (remember that the first FC was put in STEP 1, here you are putting just another FC). Train with same setup as STEP 3.
- STEP 5: Change the neurons numbers in FC layers into 1000. For regularization, use Dropout (with a rate of 0.5). Train the whole system for 40 epochs.

The traces from running testCNN.py <mode> for each of the 5 steps should be saved in output.txt, as indicated above. Each step is 1 point.

What to submit:

- Code
- A short write-up about your implementation with results and your observations from each training. Note that in each step you will train the corresponding architecture and report the accuracy on the test data. Also show how training/test loss and accuracy is varying with each iteration during the network training using plots.