CSC 249/449 Machine Vision: Homework 3

Term: Spring 2018
Instructor: Dr. Chenliang Xu
Due Date: May 1, 2018 11:59PM (Eastern Time)
Version: 1.0 (April 1, 2018)

Constraints: This assignment may be discussed with other students in the class but must be written independently. Over-the-shoulder Matlab coding is strictly prohibited. Web/Google-searching for background material is permitted, but web/google-searching for specific answers to the questions is prohibited (the lectures and readings provide nearly all of the information needed to answer these questions).

Problem 1 (total 80 points): Recognizing Handwritten Digits

This problem will get you working on classifying handwritten digits from 0-9. With the well-trained Convolutional Neural Network (CNN), you may be able to help postman recognize scribbled zip code.

Library: We will use **MatConvNet**, a MATLAB toolbox implementing CNNs for computer vision applications. To install the library, follow these steps:

1. Download the library source code from `https://github.com/vlfeat/matconvnet.git` and unpack it into a directory of your choice. Call the path to this directory `<MatConvNet>`.
2. Follow the official guide in `http://www.vlfeat.org/matconvnet/install/` to compile the library for CPU. (Note that the CPU-compiled library is good enough for you to implement HW3; however, if you want to use it for implementing non-trivial course project, you should try to compile it with GPU.)
3. Move the directory `three_m/` into `<MatConvNet>/examples/`.

Data and code: You will use the MNIST database of handwritten digits, available from `three_m/data/mnist-baseline/imdb.mat`. MNIST has a training set of 42,000 examples, a validation set of 14,000 examples, and a test set of 14,000 examples. The digits have been size-normalized and centered in a 28×28 image.

Work: You will be implementing a baseline CNN for digit classification on the original dataset, plus an advanced CNN for the *attacked* dataset, in which images are either corrupted by certain amount of noise or randomly rotated by certain angles.

1. **(25 points)** Implement the missing body of code in `cnn_init.m`, which should implement the following network structure.

   (conv1): Conv2d(in_channels=1, out_channels=8, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
   (relu1): Relu
   (pool1): Maxpool(kernel_size=(2, 2), stride=(2, 2), padding=(0, 0))
   (conv2): Conv2d(in_channels=8, out_channels=16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
   (relu2): Relu
   (pool2): Maxpool(kernel_size=(2, 2), stride=(2, 2), padding=(0, 0))
   (conv3): Conv2d(in_channels=16, out_channels=32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
   (relu3): Relu
   (pool3): Maxpool(kernel_size=(7, 7), stride=(1, 1), padding=(0, 0))
   (conv4): Conv2d(in_channels=32, out_channels=10, kernel_size=(1, 1), stride=(1, 1), padding=(0, 0))

   Refer to the official document for layer APIs in the `SimpleNN` wrapper offered by MatConvNet. Initialize W weights like $0.01 \times randn$, where $randn$ samples from a zero mean, unit standard deviation gaussian, and initialize biases as all zeros. Once finishing the network definition, you can run `mnist_train.m` to start network training. After each epoch, a checkpoint will be saved as `three_m/data/mnist/net-epoch-%d.mat`. A pretrained model is also provided in `three_m/data/mnist-baseline/net-epoch-20.mat`. Run `mnist_eval.m` to evaluate the model on test set. Typically, after 20 epochs of training, the model should achieve a test accuracy above 98%.

2. **(35 points)** Try to improve the classification performance with a modified network structure, data augmentation, network assembling or whatever you can think of. In a PDF file named `enhance.pdf`, report your way of improvement and how much it increases, compared with the accuracy achieved by the baseline CNN in (1).
3. **(20 points)** Attack the test images in the MNIST dataset by either adding Gaussian noise or rotation, or do both with the script `attack.m` (we have provided this script to you; see `mnist_eval.m` for examples of turning the parameters `sigma` and `degree`). Run `mnist_eval.m` to see how accuracy drops when testing with the networks built above (both baseline and your improved one). Implement script `plot_curve.m` to plot accuracy under different noise intensities and rotation degrees. Explain the reasons for accuracy decrease in `enhance.pdf`.

Extra Credit (20 points) Implement an enhanced CNN to recognize digits in these attacked images. Present results with accuracy plots under different noise intensities and rotation degrees. Explain how the model is improved and why it can resist noise or rotation.

Submission Process: Zip up your MATLAB code and update to the submission folder in the Blackboard assignments. The zip of your folder should expand to one-level up from the main files: the download of the files expanded to `three_m/xyz.m`, and your implementation should do the same.

Be sure to include all of the files, we will run your code. The MATLAB code needs to run on our machines. Do not include any local paths or local files. If any are used, then be sure to upload them with the other code (none are expected). Do not change output formats/locations of the scripts as these may be auto-graded by our system.

Grading and Evaluation: The credit for each problem in this set is given in parentheses at the stated question (sub-question fraction of points is also given at the sub-questions). Partial credit will be given for both paper and MATLAB questions. For MATLAB questions, if the code does not complete, then limited or no credit will be given.