Homework Assignment # 7 (We will skip #4, #5, and #6)

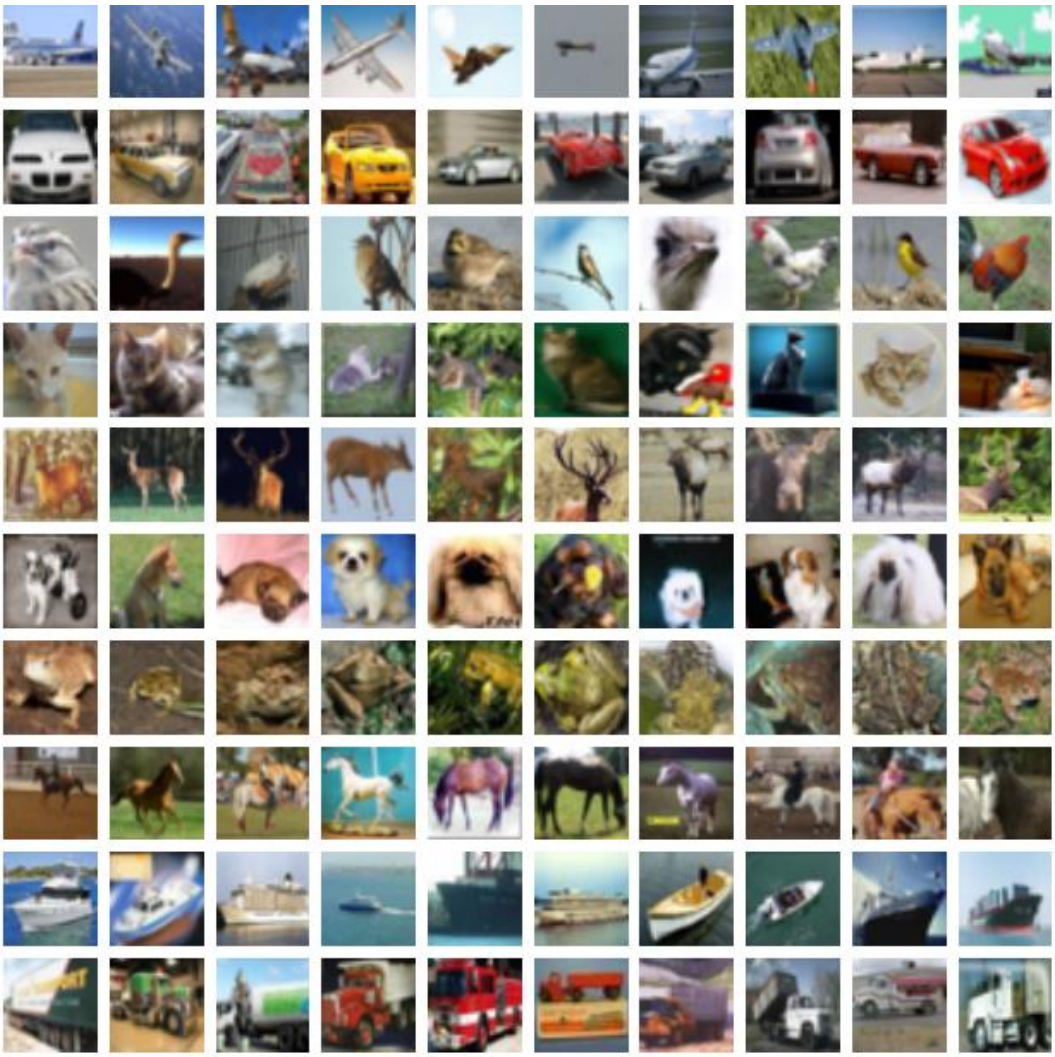
DUE: 5:00pm, Sunday March 14th (Electronic turnin required)

In this assignment, you are to experiment with using neural networks for a recognition task. An important caveat here is that you should limit your ambition and computational scope (unless you have access to powerful GPUs for your experimentation). If you use CSIL and CSTL, do remember that Tensorflow is available only in the CPU version. As the ECI technical support had so much trouble installing Tensorflow, I did not trouble them to install other packages such as PyTorch or MxNet. You might be able to get some of these packages to work on your own personal computer.

Furthermore, most "entry-level" public datasets have been experimented with extensively. Often times, it is nontrivial for you to beat published benchmark recognition rates that can be higher than 99%. However, hitting such high recognition rates is not the goal of this homework. You might be able to achieve a high recognition rate by adapting some published networks or starting the training from some published parameter sets and fine tuning further (transfer learning). But you will certainly learn more by rolling your own codes and performing training from scratch.

In terms of data sets, we will use the following:

CiFar10 https://www.cs.toronto.edu/~kriz/cifar.html: Or follow the testImages link in class website. 10 classes (airplane, auto, bird, cat, deer, dog, frog, horse, ship, truck) with 60,000 32x32 color images (6,000 per class) separated into 50,000 training and 10,000 test.



This is a “reasonably” small data set (<200MB) that hopefully will not overwhelm your CPU, GPU and disk storage. They are normalized, cropped, preprocessed and 1-hot data sets, so you do not need to perform trimming, segmentation and region proposals.

As far as networks are concerned, while you can use FNN, CNN, Resnet, or Densenet with your choice of number of blocks, number of layers, number of neurons per layer etc., it is important to limit your expectation. Large networks, such as 100-layer Desnset, can consume large storage space and run time.

Instead, you will design a simple and “shallow” network for this assignment. The following parameters are recommended by your TA Mr. Da Zhang:

Input – (CNN – Relu – pooling)1 – (CNN – Relu – Pooling)2 – (CNN – Relu - CNN – Relu – Pooling)3 – (FNN)4 – (Softmax)5

There are five major blocks concatenated sequentially, with each block comprising a varying number of modules.

- Input is a vector of size 32x32x3.

- CNN1 is 7x7x3x64, CNN2 is 3x3x64x128, and CNN31 is 3x3x128x256 and CNN32 is 3x3x256x256, (width x height x # of input channels x # output channels).
- FNN4 is of a dimension 4096x10.
- All pooling operations are maximum pool with a stride 2.

It is easily deduced that output of block 1 is 16x16x64, block 2 is 8x8x128, and block 3 is 4x4x256 (all with proper border padding). Output of the FNN blocks should be 10 channels. The Softmax function will normalize the 10-channel output from the FNN into a probability distribution for the 10 classes, which are then matched with the ground-truth, one-hot labels using cross entropy loss. The error is then back-propagated to refine the network parameters

For training, you will use the 50,000 training images with a batch size of 64 (128 and 256 should still be fine, but the default batch size of 10,000 of CiFar10 will overwhelm the CPU). Each training epoch will comprise running through all 50,000 training images (64, 128, or 256 at a time) in a random order through the backpropagation training (with weights randomly initialized for the 1st epoch). A total of 10 epochs should be attempted.

You are to output (graph) two pieces information:

(1) The training time as function of epoch (10 data points), also
(2) The training and testing errors after each epoch of training (20 data points in a single plot).

Your tasks are to write two pieces of codes: The first piece (prog7, again, we will skip #4, #5, and #6) has the following components: (1) process images (input, batch, format conversion, etc.), (2) perform training (with proper batching), (3) estimate performance (i.e., error rate and run time of each epoch) and (4) output (save) the final network parameters.

The second piece (prog72) has the following components: (1) input (load) the final network parameters from prog7, (2) process images in the test batch (10,000 images), and (3) estimate the performance (error rate) using the final network parameters on the test images only and output the error rate.

You should turn in both codes (prog7 and prog72) and graphs for runtime (training images only, 10 data points) and training and testing errors (20 data points in one graph).

Again, this exercise is not to produce top-notched results but to convince yourself that even with so many parameters, the network can gain a “reasonable” performance by learning from scratch. According to Da, it takes about 5-6 minutes for one epoch training on CSIL using CPU. So roughly you should spend an hour for 10 epochs. The network size should be less than 20MB.

If you do additional testing – like changing the network topology, adding more layers (such as batch normalization layers), using different activation functions (such as SeLu), etc. - and obtain interesting performance, please write a short README file to alert the reader what you did.