CS/ECE 181 Homework 7 (March 13)

(Upload the zip file by 5pm, Friday, March 13, on Gauchospace).

You will write a report on each of the questions below. Upload your report together with your code as a zip file to GauchoSpace. Name the main folder as "yourlastname_firstname". Organize the code into a subfolder named "code" inside the respective sub-folders.

We will be using CNNs to address the CIFAR-10 10-class classification problem.

Dataset: Dataset can be found at `https://www.cs.toronto.edu/~kriz/cifar.html` Training and testing splits are also mentioned on the webpage.

1. Visualize one sample image for each of the 10 classes.

2. Design a model to classify these 10 classes. You may choose parameters of your model architecture such as number of layers, number of neurons and the loss function from the classical MNIST model (see below)

   OR

   you may choose any other model, in which case you should comment on your choices.

   Implement your model in CNN environment that you are comfortable with (MATLAB/PyTorch/Tensorflow) and train the model weights using the training set. Plot the training and testing loss and accuracy over time and observe the network's improvement with every iteration.

3. Calculate the total number of free, trainable parameters in your model, i.e., total number of weights, biases.

4. Visualize one wrongly-classified sample from each of the 10 classes. Choose these samples from the test set.

- The following link explains about some of the numerical details relating to conv net:

  `https://cs231n.github.io/convolutional-networks/#conv`

- The following links describe the MNIST CNN which we would like to implement for the current problem:

  `https://www.tensorflow.org/tutorials/layers`

  `https://www.tensorflow.org/tutorials/quickstart/advanced`

  `https://github.com/pytorch/examples/tree/master/mnist`

  `https://www.mathworks.com/help/deeplearning/examples/create-simple-deep-lear`
  `html`

Example of the network architecture we want to implement:

- Convolutional Layer 1: Applies 32 5x5 filters (extracting 5x5-pixel subregions), with ReLU activation function

- Pooling Layer 1: Performs max pooling with a 2x2 filter and stride of 2 (which specifies that pooled regions do not overlap)
- Convolutional Layer 2: Applies 64 5x5 filters, with ReLU activation function
- Pooling Layer 2: Again, performs max pooling with a 2x2 filter and stride of 2
- Dense Layer 1: 1,024 neurons, with dropout regularization rate of 0.4 (probability of 0.4 that any given element will be dropped during training)
- Dense Layer 2 (Logits Layer): 10 neurons, one for each digit target class (0–9).