HW 5 Due: Thursday, May 2, 2023 at 2:00 pm.

Please hand in this Homework as follows:
• Upload to Gradescope `HW5 programming`: A `ipynb` file for both problems.

1. *Convolutional networks for MNIST* (40 points) The code in this notebook allows you to train a particular convolutional neural network (which we call the *original model*) on MNIST data. When you click on the link it will have a menu button `open with`. Use the Google Colab option. The code also saves the model in your directory and allows you to reload the model and continue training or to simply test the model on a new data set. You have two options:

   - You can download the notebook into your computer.
   - Or, you can decide to use Google Colab together with Google Drive. Install Google Drive on your computer. It will create a folder called Google Drive in your home directory. Any file you put in there will automatically be synced into your Google Drive. You can move your data files into Google Drive so that when you are running Colab it will see those files. Save the notebook linked above by going to the `file` menu and choosing `save a copy in drive`. This will be saved in your Google Colab folder (A subfolder of your drive). Upload the MNIST data to your Google Drive and you're ready to go. The advantage is that you can activate a GPU and have the algorithm run very fast. Once you open you own Colab notebook, use the `runtime` menu, choose the `Change runtime type` and pick gpu from the dropdown menu.

   (a) Compute the total number of parameters in the original model. And run this model. You shouldn't run more than 20 epochs. (On my macbook-pro it takes about 90 seconds per epoch with all the training data.) You can do this with only 10000 training data to expedite the experiments. For each experiment plot the error rate on training and validation as a function of the epoch number.

   Show an image with the 32 5×5 filters that are estimated in the first layer of the model

   (b) Experiment with changing parameters of the network:

      i. Keep the same number of layers and change layer parameters reducing number of parameters by half in one experiment and doubling the number parameters in another. Try a few different options. Report the results.

      ii. Design a deeper network with approximately the same number of parameters as the original network. Report the results.

      iii. Once you pick the best configuration try it on the full training set and report the result

   (c) Handling variability. A randomly shifted test set is been created in the function `get_mnist_trans` by taking each digit and applying random shift sampled uniformly between $+/-$ `shift/2` pixels in each direction. For different sizes of `shift` Display a few of these examples alongside the original digits.

Using the original trained network to test on this data set. Show the classification rate as a function of `shift`.

Try to propose changes to the network architecture so that still training on the **original training** set you would perform better on the transformed test set. Perform some experiments using a transformed validation set and show the final results on the transformed test set.

2. *Convolutional networks for CIFAR10* (40 points)

   (a) Read in the cifar data. The files are in `CIFAR/`. There is a function to read the cifar data in the notebook. Display some of the images.

   (b) Modify the code to apply the original network to the cifar data. Remember that the images now have 3 color channels. Again plot training and validation error against epoch number. Plot the first layer filters.

   (c) Try to define a deeper network and see if you get an improvement.

   (d) Handling variability. Use `skimage.color.rgb2hsv` to transform the rgb color map of the input images to and hsv - hue, saturation, value. You can use `hsv2rgb` to transform back. (To read more about different color coding methods see: `https://en.wikipedia.org/wiki/HSL_and_HSV`. Saturation is a value between 0 and 1 providing a sense of how 'colorful' the pixel is.) For each image in the test set multiply the saturation of all pixels values by a fixed value drawn randomly between .75 and 1.25. Then convert back to RGB. Show some of the resulting images. Run the model on the modified data and report the result.