Due date: Jun. 9, 2022, 11:59 PM (KST). You have **two** late days — use it at as you wish. Once you run out of this quota, the penalty for late submission will be applied. You can either use your late days quota (or let the penalty be applied). **Clearly indicate** at the top of your submission if you seek to use the quota.

What to turn in:

1. Your submission should include your complete code base in an archive file (`zip, tar.gz`) and `q1/` and so on), and a clear README describing how to run it.
2. A brief report (typed up, submit as a PDF file, NO handwritten scanned copies) describing what you solved and implemented and known failure cases. The report is **important** since we will be evaluating the scores mostly based on the report. Attach **screenshots of the code** or **copy and paste the code using latex function(ex.lstlisting) which preserves the code style.** (10 points deducted if code is not attached)
3. Submit your entire code and report to PLMS.

Notes from instructor:

- Start early!
- You may ask the TA or instructor for suggestions, and discuss the problem with others (minimally). But **all parts of the submitted code must be your own**.
- **Asking questions via Q/A board highly recommended**.
- Use Python for your implementation.
- This time, you can freely use existing libraries, Keras and Pytorch (you cannot use TensorFlow1).

Instructor: W. H. Kim (wonhwa@postech.ac.kr),
TA: Juseong Park (wntjd9805@postech.ac.kr), Jieun Oh (jieunoh@postech.ac.kr)

Problem 1

(Artificial Neural Network, **100pts**) Design an Artificial Neural Network (ANN) using your choice of existing lirary. We will use MNIST data which is a set of hand-written digit images of numbers from 0 to 9. Pixel values are feature x and the number on each image is the class label y. The data can be downloaded from `http://yann.lecun.com/exdb/mnist/`. Make sure that you normalize the images, i.e., each pixel value should lie in $[0, 1]$.

1. (**30pt**) Design a single layer neural network that classifies y based on x. The input will be a vectorized image of dimension 784 (i.e., 28 x 28), and you will need 10 output nodes for each class. At the output nodes, use *softmax* function as the activation function and use *multi-class cross-entropy* to define the loss function. Use 32 images per batch to train your network. Try learning rates of 0.001 and 0.1 to train the network using training set, and run 100 epochs. You can use adaptive momentum optimizer (Adam) for backpropagation. Plot the loss function with respect to epoch, and test the trained model on the test set. How many epoches did it take for the model to converge and what are the training and testing accuracies?

2. (**10pt**) Design a multi-layer neural network that classifies y based on x. The settings are the same as above, but introduce a hidden layer with 128 hidden nodes. Use sigmoid as the activation function for the hidden nodes. Now fix the learning rate as 0.001 and train the network for 100 epochs. Plot the loss function with respect to epoch. How many epoches did it take for the model to converge and what are the training and testing accuracies? Did it get better than last time?

3. (**10pt**) Design a multi-layer neural network that classifies y based on x. This time, introduce two hidden layers with 128 and 4 hidden nodes respectively, and the rest of the settings should be the same as above. What are the training and testing accuracies? Did it get better than last time?

4. (**20pt**) From the setting above, when you do the testing, save the activated values from the 4 hidden nodes and take average of each node per class and report them. For example, if there are 100 images in the test set that are classified as 0, take the average across the 100 images per node and report 4 bits of numbers. Do you see any pattern across different classes?

5. (**10pt**) Design a multi-layer neural network that classifies y based on x. This time, introduce two hidden layers with 128 and 2 hidden nodes respectively, and the rest of the settings should be the same as before. What are the training and testing accuracies? Did it get better than last time? Why is this happening?

6. (**20pt**) In the above setting(5.), for the training samples, take the activated values from the 2 hidden nodes across the entire images and plot them. You should be able to visualize each sample in trained 2D feature space as a scatter plot. Color code each sample, i.e., different color should assigned for different classes in the scatter plot(total 10 colors). What do you see? What are the classes that cannot be discrimiated correctly?