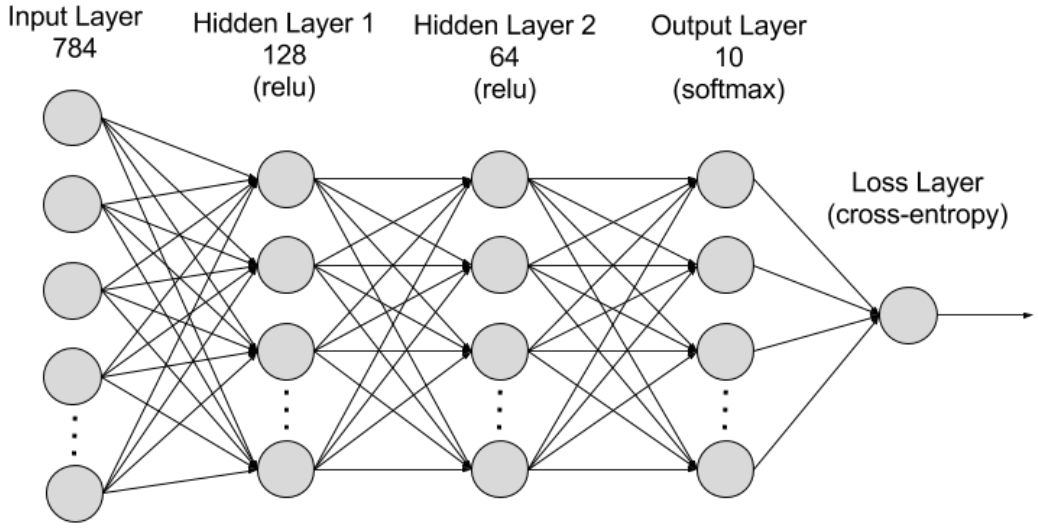
Statistical Learning Theory,

Assignment 2 – Using neural networks to recognize handwritten digits

Due: 11:59PM, November 22, 2020
Issued: November 8, 2020

Instructions:

- Implement different fully connected neural network architectures and apply to classify handwritten MNIST digits.
- Feel free to use the codes in "Neural_Networks_MNIST.ipynb" distributed during the class, and modify them to complete this assignment.
- For fully connected neural networks, there are three essential components that define the network's architecture including **# layers, # nodes in each of the layers, activation function**, used at each of those layers. This assignment explores the first two key components (layer #, node #). While there are some rules of thumb, the only way to determine the best architecture for any particular task is ***empirically***. Sometimes the "sensible defaults" will work well, and other times they won't work at all. The only way to find out for sure if your neural network works on your data is to test it, and measure your performance.
- For instance, the neural networks shown below have 2 different hidden layers, and the first and second hidden layers have 128 and 64 nodes, respectively.



- You may **not** use any other languages other than Python to accomplish your task.

Questions:

- For questions (1 and 2), use the hyper-parameter values and learning-settings given as follows: SGD optimizer with learning rate=0.05, epochs=10, activation function=ReLU, loss function=softmax + cross-entropy, training batch size=100. In case of (3), freely adjust the above hyper-parameter values.
-

1. **Performance over different layer width**: Assume you have a multi-layer perceptron (also know as a fully connected neural network) with **one hidden layer**. Let's experiment with layers of different width (**node #**) to see how the width of a network impacts its performance. FYI, input (with 784 nodes) and output (with 10 nodes) layers are not included in the number of hidden-layers.
   A. Show your 5 different plots of training & validation accuracy with different node # [50, 100, 500, 1000, 2000] in the single hidden layer.
   B. Based on the derived results, briefly analyze performance over different layer width verbally. (*e.g., do you see overffting/underfitting? Describe the performance changes over different node #.*)

2. **Performance over different layer depth**: Assume you have a multi-layer perceptron with differently # of hidden layers. **The # of nodes in all hidden layers is the same as 500**. Let's experiment with **different # of hidden layers** with 500 nodes in each of the hidden layers to see how the depth of a network impacts its performance.
   A. Show your 5 different plots of training & validation accuracy with different # of hidden layers [1, 2, 3, 4, 5].
   B. Based on the derived results, briefly analyze performance over different layer depth verbally.

3. **Optimizing the neural networks beyond 97.5%**
   A. Optimize the network so that the validation accuracy exceeds 97.5%.
   B. What efforts have you made to improve accuracy of your final model?

Submission:

- Write the following items in a pdf document and submit the pdf file to the Cyber.ewha.ac.kr assignment dropbox by the deadline.
- Submit two files including (1) your source codes (format: ipynb) and (2) your answers (format: pdf). Please upload only the two files.

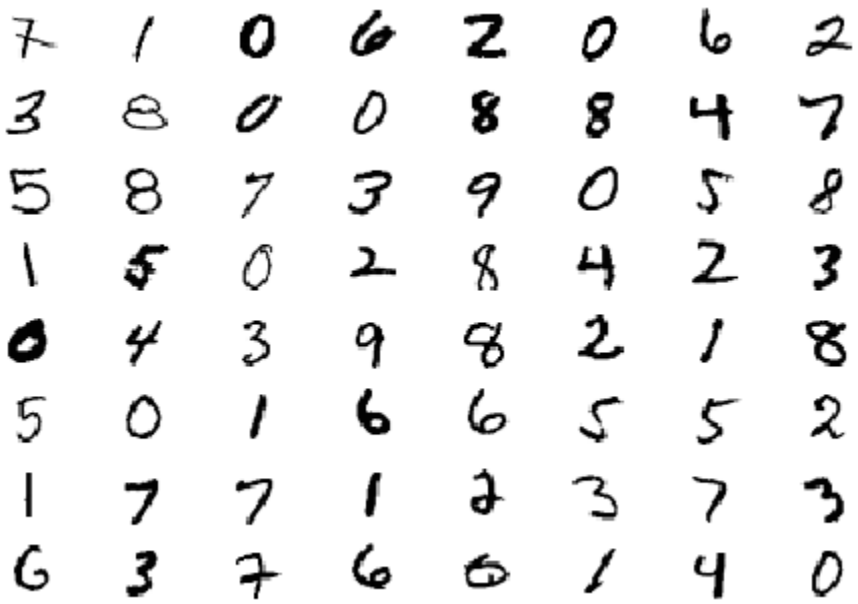
Grading:

- We will review and comment on your submission regarding the style of your Python/PyTorch code. You must attempt every question in order to receive credit.

Note:

- If you submit the assignment late, we will deduct the assignment score by 20% per day and will not accept submissions after the solution has been distributed. The solution will be uploaded 2 days after the deadline.

Dataset information:



A random selection of MNIST digits.

- The MNIST database of handwritten digits (https://en.wikipedia.org/wiki/MNIST_database, http://yann.lecun.com/exdb/mnist/), available from this page, has a training set of 60,000 examples, and a validation (or test) set of 10,000 examples. It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.