ECE 491 Homework #4

Our weekly learning objectives are:

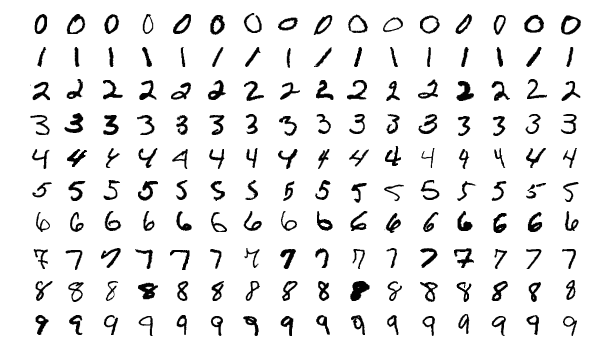
1. Construct a multilayer network (LO3).
2. Implement a Multilayer Network using Python or MATLAB (LO3).
3. Identify the contribution of model selection, underfitting and overfitting in model effectiveness (LO1).
4. Compute Forward Propagation, Backpropagation and Computational Graphs to train neural networks  (LO3, LO4)
5. Minimize chances of overfitting through “Dropout” (LO2).
6. Interpret the numerical stability and initialization issues during training phase (LO1, LO3).

In this task, you will design a 3-fully-connect-layer-network using TensorFlow-Keras or other frameworks you are familiar with (WLO1-WLO6). You will use the popular Modified [National Institute of Standards and Technology](https://en.wikipedia.org/wiki/National_Institute_of_Standards_and_Technology) database ([MNIST database](https://en.wikipedia.org/wiki/MNIST_database)). It is a collection of 70000 handwritten digits split into training and test set of 60000 and 10000 images respectively. You will do handwritten digit classification on MNIST dataset:

<http://yann.lecun.com/exdb/mnist/>

or

<https://www.kaggle.com/datasets/hojjatk/mnist-dataset>



Four files are available on LeCun’s site:

[train-images-idx3-ubyte.gz](http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz):  training set images (9912422 bytes)  
[train-labels-idx1-ubyte.gz](http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz):  training set labels (28881 bytes)  
[t10k-images-idx3-ubyte.gz](http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz):   test set images (1648877 bytes)  
[t10k-labels-idx1-ubyte.gz](http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz):   test set labels (4542 bytes)

You need to load the raw files instead of using the data in the framework. So, you **cannot** use

tf.keras.datasets.mnist.load\_data()

to load the dataset.

If you don’t know how to load the raw files, you can use the following code:

import gzip

import os

import struct

import numpy as np

def loadMNIST(DATASET\_DIR = './'):

MNIST\_TRAIN\_IMS\_GZ = os.path.join(DATASET\_DIR, "train-images-idx3-ubyte.gz")

MNIST\_TRAIN\_LBS\_GZ = os.path.join(DATASET\_DIR, "train-labels-idx1-ubyte.gz")

MNIST\_TEST\_IMS\_GZ = os.path.join(DATASET\_DIR, "t10k-images-idx3-ubyte.gz")

MNIST\_TEST\_LBS\_GZ = os.path.join(DATASET\_DIR, "t10k-labels-idx1-ubyte.gz")

print("Unpacking training images ...")

with gzip.open(MNIST\_TRAIN\_IMS\_GZ, mode='rb') as f:

magic\_num, train\_sz, nrows, ncols = struct.unpack('>llll', f.read(16))

print("magic number: %d, num of examples: %d, rows: %d, columns: %d" %

(magic\_num, train\_sz, nrows, ncols))

data\_bn = f.read()

data = struct.unpack('<'+'B'\*train\_sz\*nrows\*ncols, data\_bn)

train\_ims = np.asarray(data)

train\_ims = train\_ims.reshape(train\_sz, nrows\*ncols)

print("~"\*5)

print("Unpacking training labels ...")

with gzip.open(MNIST\_TRAIN\_LBS\_GZ, mode='rb') as f:

magic\_num, train\_sz = struct.unpack('>ll', f.read(8))

print("magic number: %d, num of examples: %d" % (magic\_num, train\_sz))

data\_bn = f.read()

data = struct.unpack('<'+'B'\*train\_sz, data\_bn)

train\_lbs = np.asarray(data)

print("~"\*5)

print("Unpacking test images ...")

with gzip.open(MNIST\_TEST\_IMS\_GZ, mode='rb') as f:

magic\_num, test\_sz, nrows, ncols = struct.unpack('>llll', f.read(16))

print("magic number: %d, num of examples: %d, rows: %d, columns: %d" %

(magic\_num, train\_sz, nrows, ncols))

data\_bn = f.read()

data = struct.unpack('<'+'B'\*test\_sz\*nrows\*ncols, data\_bn)

test\_ims = np.asarray(data)

test\_ims = test\_ims.reshape(test\_sz, nrows\*ncols)

print("~"\*5)

print("Unpacking test labels ...")

with gzip.open(MNIST\_TEST\_LBS\_GZ, mode='rb') as f:

magic\_num, test\_sz = struct.unpack('>ll', f.read(8))

print("magic number: %d, num of examples: %d" % (magic\_num, train\_sz))

data\_bn = f.read()

data = struct.unpack('<'+'B'\*test\_sz, data\_bn)

test\_lbs = np.asarray(data)

print("~"\*5)

return train\_ims, train\_lbs, test\_ims, test\_lbs

train\_ims, train\_lbs, test\_ims, test\_lbs = loadMNIST('./')

You will train the network on the training set and test it on the test set. Images in MNIST are 28x28 gray-scale image, so you may reshape the image to 784-legth vector, then build 3-fully-connect-layer-network.

Therefore, the model should be

model = tf.keras.models.Sequential([

tf.keras.layers.Dense(…),

tf.keras.layers.Dense(…),

tf.keras.layers.Dense(10, …)

])

if the input is 784-length vectors, or

model = tf.keras.models.Sequential([

tf.keras.layers.Flatten(input\_shape=(28, 28)),

tf.keras.layers.Dense(…),

tf.keras.layers.Dense(…),

tf.keras.layers.Dense(10, …)

])

if the input is 28x28 images.

You need to design each layer by yourself. You don’t have to use other type of layers in this task (like Dropout).

You may view Lab 4 at page 15--17 of file “Python Labs July Version.pdf” for more hints.

In your report, you should include:

1. (WLO4) the accuracy on the testing dataset, and the confusion matrix, and
2. (WLO1, WLO2) your network structure.
3. (WLO3) What loss function do you use (MSE, cross-entropy, etc)?
4. (WLO5, WLO6) What training parameters did you use? What kind of optimizer did you use and what is the initial learning rate? (WLO5) Did you use “dropout”? If yes what are the implications of the dropout

Append your code at the end of your report. Please submit a single pdf file.