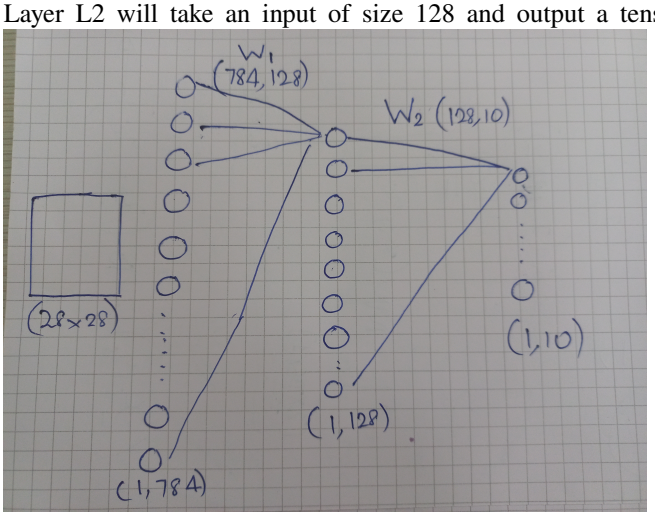
ID6001W: Applied Deep Learning
Programming Homework 2

To be submitted by March 10th, 2023 4:00 pm

1 Artificial Neural Networks (10)

1. This entire assignment must be completed in a jupyter notebook. Only numpy and matplotlib libraries can be used
2. In this assignment our goal is to code an artificial neural network including backpropagation and gradient descent from scratch.
3. The dataset to be used is the MNIST dataset. The dataset can be loaded with the given dataloading code. You can use 20% of the train data from the given script for validation. You can create batches for training if you would like, but it is not necessary.
4. The network must be a simple multi-layer perceptron of 2 layers L1 and L2. L1 layer will take an input of size(28*28=784) which is the size of an MNIST image. It will output a hidden acivation of size 128. Layer L2 will take an input of size 128 and output a tensor of size 10 with SOFTMAX class probabilities.



5. The activation function used for the hidden layer must be the ReLU activation function. The activation for the output layer must be the softmax activation. Softmax is an activation function used for multiclass classification to generate class probabilities.

$$softmax(z)_i = \frac{exp(z_i)}{\sum_j exp(z_j)}$$

6. Intialise the weights for the layers using np.random.uniform and use the Mean Squared Error as the loss function.
7. Calculate the error and the weight updates using the Chain rule of differentiation. Calculating the derivaive of the relu and the softmax activation is left to the student. Include the correct derivative in a text box.
8. In a text box write the full update rule for error of L1 and L2.
9. The weights are expected to be updated with the stochastic gradient descent algorithm. This can look like the following:

```
L1 = L1 - lr*updateL1
L2 = L2 - lr*updateL2
```

where lr is the learning rate and L1 and L2 are the weight matrices for the network.

10. Calculate and store the loss and accuracy in an array. Both training loss and validation loss must be calculated. Once training is complete, plot the array in matplotlib with loss and accuracy on the y axis and epochs on the x axis. Accuracy should be in yellow between 0 to 1 and loss should be in blue. Marks will only be provided if plots are provided. Finally there must be two graphs. One for training and one for validation. Graphs should be well labeled and legend must be provided.

11. Optional Guide for functions:
relu(x) -> Ouputs: ReLU activation of x
d_relu(x) -> Outputs: Derivative of relu for chain rule
d_softmax(x) -> Outputs: Derivative of softmax for chain rule
softmax(x) -> Ouputs: Softmax activation of x
forward_and_backward(x, y) -> Outputs: out, UpdateL1, UpdateL2
loss(out, y) -> Outputs: Mean Squared Error between out and target y
train(x, y, lr, epochs) -> Your training loop and gradient descent update

```
def fetch(url):
  import requests, gzip, os, hashlib, numpy
  fp = os.path.join("/tmp", hashlib.md5(url.encode('utf-8')).hexdigest())
  if os.path.isfile(fp):
    with open(fp, "rb") as f:
      dat = f.read()
  else:
    with open(fp, "wb") as f:
      dat = requests.get(url).content
      f.write(dat)
  return numpy.frombuffer(gzip.decompress(dat), dtype=np.uint8).copy()
X_train = fetch("http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz")
    [0x10:].reshape((-1, 28, 28))
Y_train = fetch("http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz")[8:]
X_test = fetch("http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz")
    [0x10:].reshape((-1, 28, 28))
Y_test = fetch("http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz")[8:]
```