Problem 4.

Develop and train a deep multi-layered perceptron (MLP), to classify hand-written digits from MNIST dataset

1. **Shape of the training samples**: The shape of each training image is [1, 28, 28]), i.e 28 x 28 for height and width and 1 channel for grayscale.

2. **Splitting ratio**: The train data has 60,000 images and the test has 10,000. So the splitting ratio is 85:15. As the total images are 70,000, (60000/70000 *100) and (10000/70000 *100).

3. **How every layer of MLP is represented mathematically**: There are 4 layers in total. One input layer, 2 hidden layers and one output layer. The pixels in the 28X28 handwritten digit image are flattened to form an array of 784-pixel values. The function of the input layer is just to pass-on the input (array of 784 pixels) into the first hidden layer.

   The first hidden layer has 120 neurons that are each fed the input array. After calculating the result from (Wx + b) where 'W' are the weights fed into each 'x' neuron and 'b' is the bias. Each neuron generates an output that is fed into each neuron of the next layer. The next layer has 84 neurons and takes 120 inputs from the previous layer. The output of this layer is fed into the last layer which is the Output Layer.
   The Output Layer has only 10 neurons for the 10 classes that we have(digits between 0-9). For the output layer, we use the softmax function. Softmax takes the output of the last layer(called logits) which could be any 10 real values and converts it into another 10 real values that sum to 1. Softmax transforms the values between 0 and 1, such that they can be interpreted as probabilities. The maximum value pertains to the class predicted by the classifier.

4. **Non-linear activation function**: The non linear activation function used is **ReLU** .
   f(x) = max(0,x).
   ReLu function does not activate all the neurons at the same time and clips all the negative values and keeps the positive values just the same.

5. **Loss function and Optimizer**: **Cross Entropy Loss** is used as our loss function.Cross Entropy is a measure of uncertainty between the classes. Cross Entropy could be used to identify the negative log likelihood of a SoftMax/Bernoulli distribution. In other terms, any loss consisting of a negative log-likelihood is a cross entropy between the empirical distribution of the training set and the probability distribution defined by the model. It is nothing but a log loss which measures the performance of models in classification tasks because its output probability value is between 0 and 1. As we diverge from the actual labels, cross entropy loss increases.

f(x)= CrossEntropy(y,x)=−∑ yclog($S(x)c$)= ∑c yclog(pc)

Optimizers are used to minimize loss as we move ahead through our training. For my optimizer, I have used the **Adam** optimizer. And set the learning rate as 0.01. ADAM (ADAptive Moment Estimation) optimizer, which is most common nowadays as it is a combination of RMSprop and Stochastic Gradient Descent with momentum. Adaptive Moment Estimation, (ADAM) uses the estimations of the first as well as second moments of the gradient to adapt the existing learning rate for each weight of the neural network. It uses squared gradients to scale the learning rate (RMSprop) in this case and moving average of the gradient.

6. I have trained the model for 10 epochs and achieved a Train accuracy of Train accuracy of 99.28 and a Test accuracy of 97.730% .