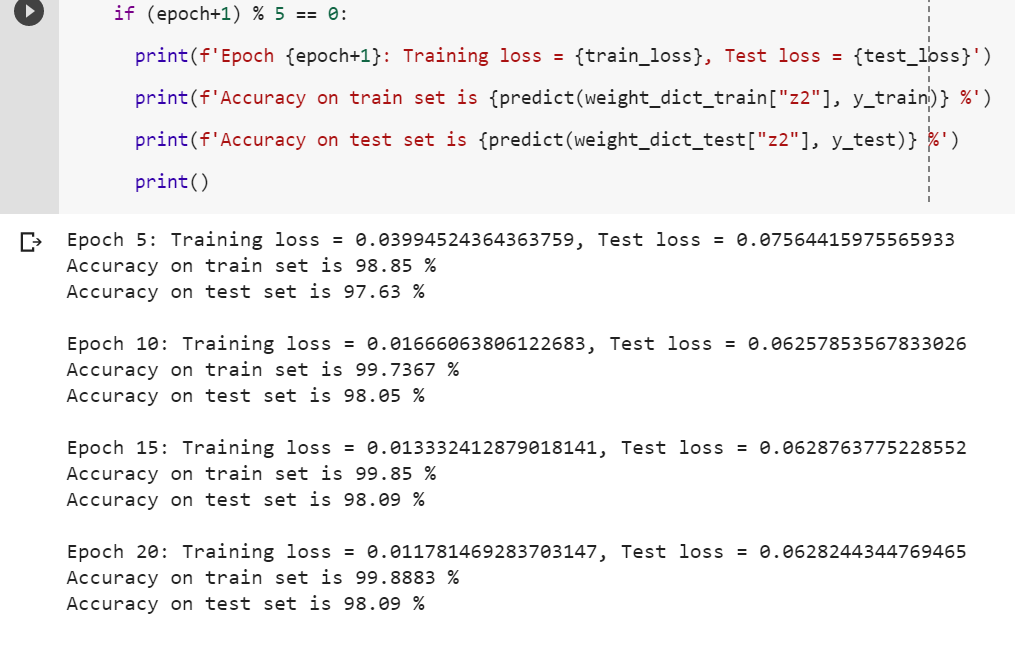
**Sayantan Dutta**

I have achieved accuracy of 98.09% in test data. Below is the screenshot:



The MNIST dataset was prepared. One-hot encoding was used for this case of multi-class output. Train and test data were separated. The train data was shuffled.

The hyper-parameters used:

initial learning rate (lr) = 0.5,  
no. of epochs (for training) = 20,   
number of hidden layers = 1  
hidden units (in the hidden layer) = 100  
batch size = 16  
  
Other parameters:

input units = 784 (28 \* 28)  
output units = 10

Weights and biases are initialized randomly for each neuron. The variance in the initialization of the neurons is scaled by dividing with the square root of the number of inputs to a neuron.

Dictionary ‘parameter\_dict’ stored the weights and biases values of the initial parameters.

Sigmoid function is used as activation function.   
Function name: ‘**sigmoid\_function**’

Negative log likelihood or Cross-Entropy function is used as loss function.  
Function name: ‘**calculate\_loss**’  
  
During the training of the neural network, in the feed-forward process dictionary ‘weight\_dict’ stores the weights and biases as the network predicts the outputs.  
Function name: ‘**feedforward**’

In the back-propagation process, Mini-batch Stochastic Gradient Descent process is used.  
Function name: ‘**backpropagation’**

Dictionary ‘gradients’ stored the gradient value. Mini batches of size 16 is used over many iterations to train the network within each epoch. Parameters are updated after each mini batch is used for feedforward, backward process. The training/testing loss and accuracy are calculated after completion of each epoch.

The initial learning rate is set to 0.5 and is kept unchanged till 30% of the total epochs are completed for training. When the network training is between 30% and 80% of the total epochs the learning rate is reduced to 0.1. Once the network is trained above 80% of the total epochs the learning rate is reduced to 0.01.