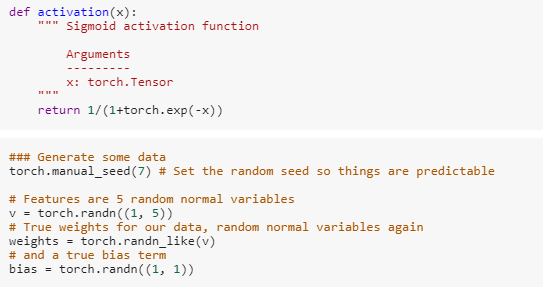
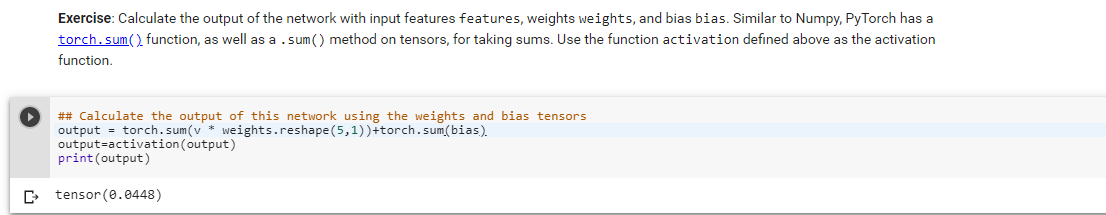
Exercise 1 :

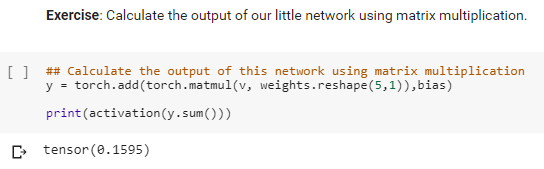


Randomly initialized the weight and bias values and used torch.sum for sum the values and \* for multiplication as shown below



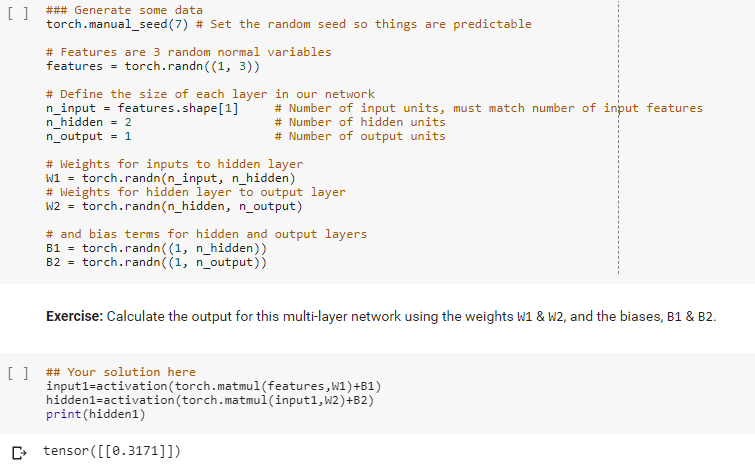
Exercise 2 :

Used torch.matmul function to do matrix multiplication and torch.sum for sum the values



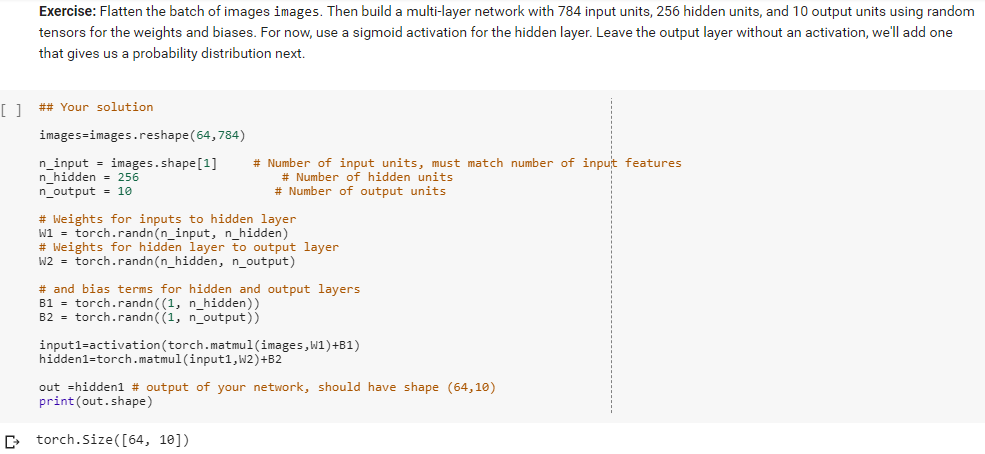
Exercise 3 :

Used torch.matmul function for matrix multiplication and activation function to get the output in range of 0-1 or -1 to 1. The output is stored in the hidden1 variable.



Exercise 4 :

Flattened the images using the function reshape();  
created random weight and bias for input of size 784 and 256 and did matmul of input and W1 and sum with bias to get hidden layer and send that to sigmoid activation function and did the same for hidden layer and W2, B2 to get the output layer

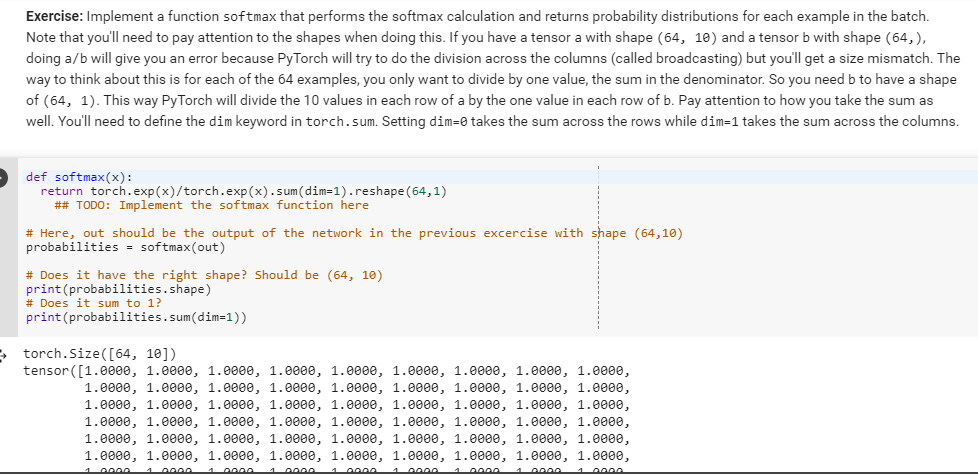


Exercise 5:

Softmax function is the ratio of exponential of x to the total sum of all the exponential of x

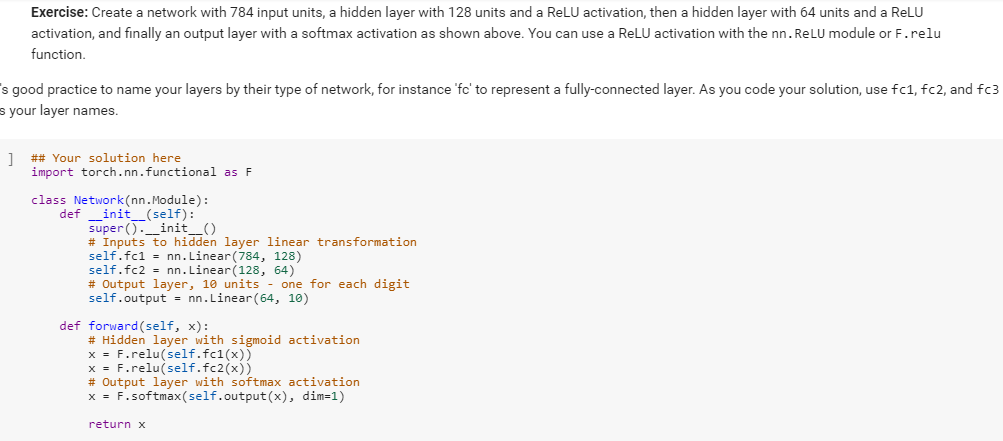
i.e, torch.exp(x)/torch.exp(x).sum(dim=1).reshape(64,1)

used dim=1 to perform sum operation on column

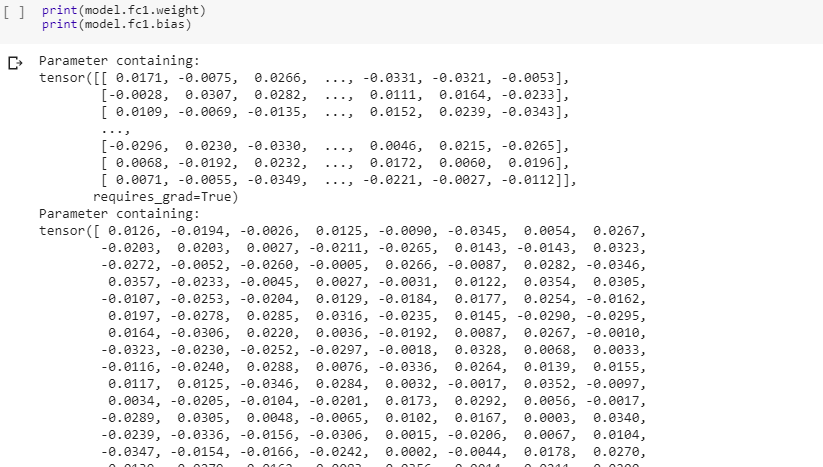


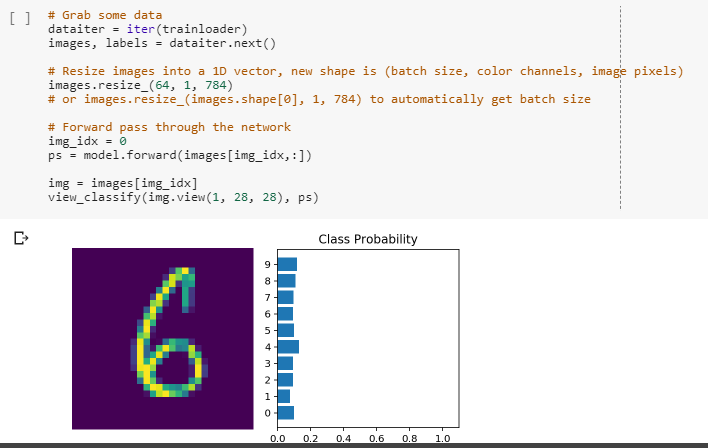
Exercise 6:

nn.linear() is th function used to perform linear transformation and used ReLu(Rectified Linear Unit) activation function (f.Relu) and for output layer used softmax activation function

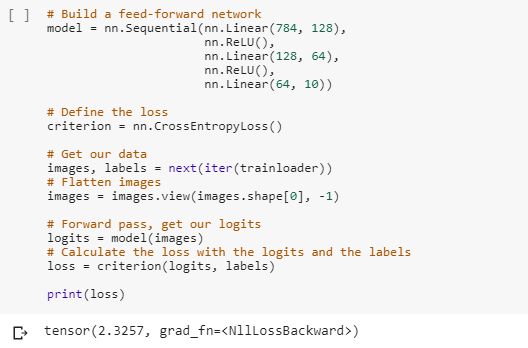


And the model’s weight and bias looks like follow,





nn.sequential is used to pass the tensors in the network in a sequence and we can used orderDict to specify the name for the layers uniquely.



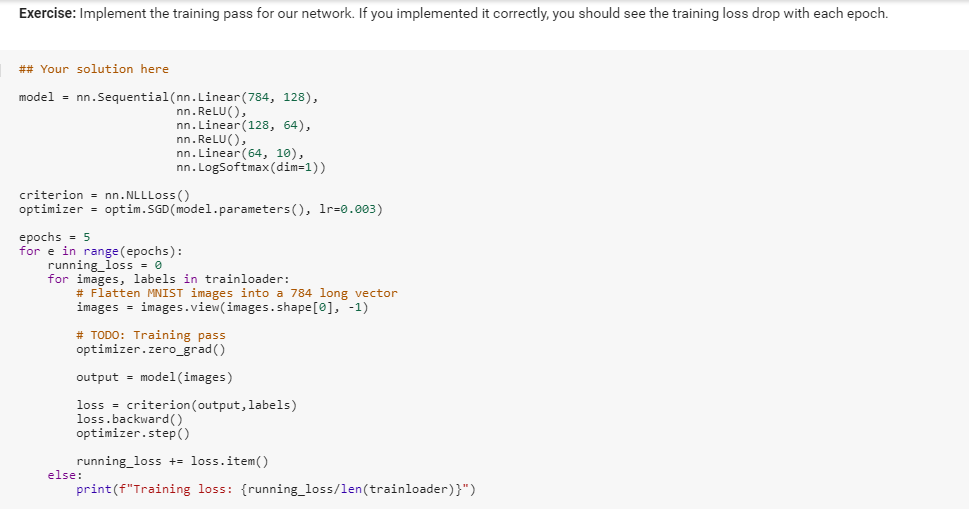
Used log-softmax function for output and used NLLLoss() loss function to calculate the loss



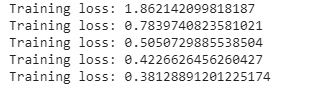
AutoGrad is used to perform backpropagation and we used stochastic gradient descent optimizer

Calculated the loss using NLLLoss() function and backpropagation is done using the loss.backward()

And used optm.SGD() to reduce the loss by correcting the learned values in backward

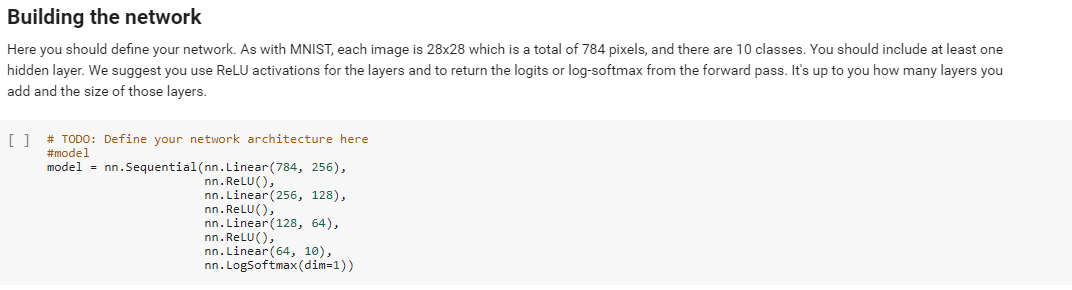


You can see below the loss is getting reduced for each epoch,

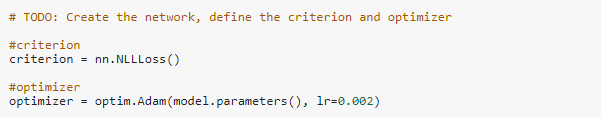


Fashion MNIST:

I have created a network with 3 hidden layers (256 units, 128 units and 64 units) and output layer of 10 units and used relu activation function for hidden layers and logSoftmax for output layer



I have used loss function NLLLoss() and Adam optimizer with learning rate of 0.002



I have trained the model for 5 epochs and for each epoch I calculated the loss function and backpropagated using loss.backward() and optimized the values learnt in previous layers using adam optimizer to reduce the loss of the DNN. You can see the loss gets reduced for each epoch

