Answer 2:

1. We have implemented an autoencoder using fully connected layers for MNIST dataset.
    - The encoder will have 2 layers (with 256, and 128 neurons) and the decoder will also have two layers (with 256 and 784 neurons)
    - Trained this network using MSEloss
    - For 10 epochs
    - Optimizer used is SGD

    Steps followed:

    I. Input size is 28 X 28 and 1 is the depth of the given image = 1 * 28 * 28

    II. Flatten(1*28*28)

        a. Flatten operation that converts the feature volume to 1D feature vectors which is passed to FC layer.

    III. First Encoding Fully Connected layers with input neurons = 784 and hidden neurons = 256

        self.fcEncoder1 = nn.Linear(784, 256);

    IV. And then apply RELU Activation function for linear layers

    V. Second Encoding Fully Connected layers with input neurons = 256 and hidden neurons = 128

        self.fcEncoder2 = nn.Linear(256, 128);

    VI. And then apply RELU Activation function for linear layers

    VII. First Decoding Fully Connected layers with input neurons = 128 and hidden neurons = 256

        self.fcDecoder1 = nn.Linear(128, 256);

    VIII. And then apply RELU Activation function for linear layers

    IX. Second Decoding Fully Connected layers with input neurons = 256 and hidden neurons = 784

        self.fcDecoder2 = nn.Linear(256, 784);

    X. Lastly, apply Sigmoid Activation function for linear layers to get the output

    XI. Here, for Training parameters used

        - Relu activation function
        - SGD is used as an optimizer
        - learning rate of 1.0
        - 10 epochs
        - mini-batch size of 10
        - MSEloss

Results:

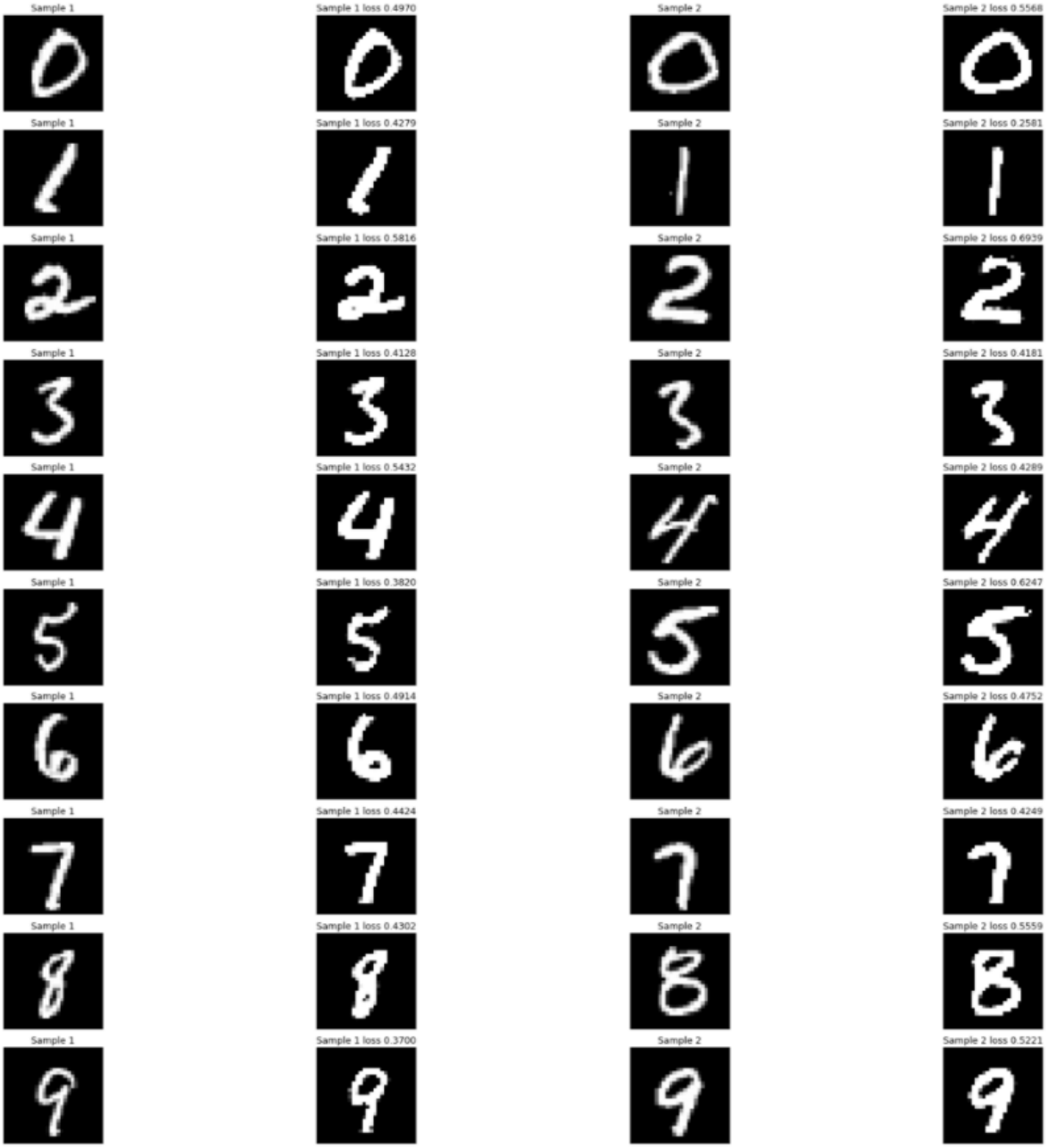
Total parameters is = 468368

Encoder parameters is = 233856.0

Decoder parameters is = 234512.0

Average loss calculated for 10 epochs is = 0.4988160799741745

The 20 sample reconstructed images from testing data in the report (2 image for each class) along with the original images are:



GitHub Link:

https://github.com/SunilDevlops/Programs/blob/master/ComputerVision/ProgramAssignment2/Question2/MNIST_Dataset_AutoEncoder_FC.ipynb

2. We have also implemented a convolution autoencoder for MNIST dataset.
   - The encoder will have 2 convolution layers and two max-pool layers followed by each convolutional layers
   - Used kernel size 3x3, relu activation, and padding of 1 to preserve the shape of the input feature map
   - And, then used the decoder to have three convolutional layers with kernel shape 3x3 and padding of 1 to preserve the feature map shape
   - Also used Upsampling layer
   - Trained this network using MSEloss
   - For 10 epochs
   - Same Optimizer used as above i.e., SGD

   Steps followed:

   I. Input size is 28 X 28 and 1 is the depth of the given image
      = 28 * 28 * 1
   II. First 2D convolutional Encoder layer, taking in 1 input channel (image), outputting 32 convolutional features, with a square kernel size of 3 and padding 1
      = 28 * 28 * 32
   III. Apply RELU
   IV. Max pooling of 2 * 2
      = 14 * 14 * 32
   V. Second 2D convolutional Encoder layer, taking in 32 input channel (image), outputting 64 convolutional features, with a square kernel size of 3 and padding 1
      = 14 * 14 * 64
   VI. Apply RELU
   VII. Max pooling of 2 * 2
      = 7 * 7 * 64
   VIII. First 2D convolutional Decoder layer, taking in 64 input channel (image), outputting 64 convolutional features, with a square kernel size of 3 and padding 1
      = 7 * 7 * 64
   IX. Apply RELU
   X. Used Upsampling layer

= 14 * 14 * 64

XI. Second 2D convolutional Decoder layer, taking in 64 input channel (image), outputting 32 convolutional features, with a square kernel size of 3 and padding 1

= 14 * 14 * 32

XII. Apply RELU

XIII. Used Upsampling layer

= 28 * 28 * 32

XIV. Third 2D convolutional Decoder layer, taking in 32 input channel (image), outputting 1 convolutional features, with a square kernel size of 3 and padding 1

= 28 * 28 * 1

XV. Lastly, apply Sigmoid Activation function for linear layers and flatten to get the output

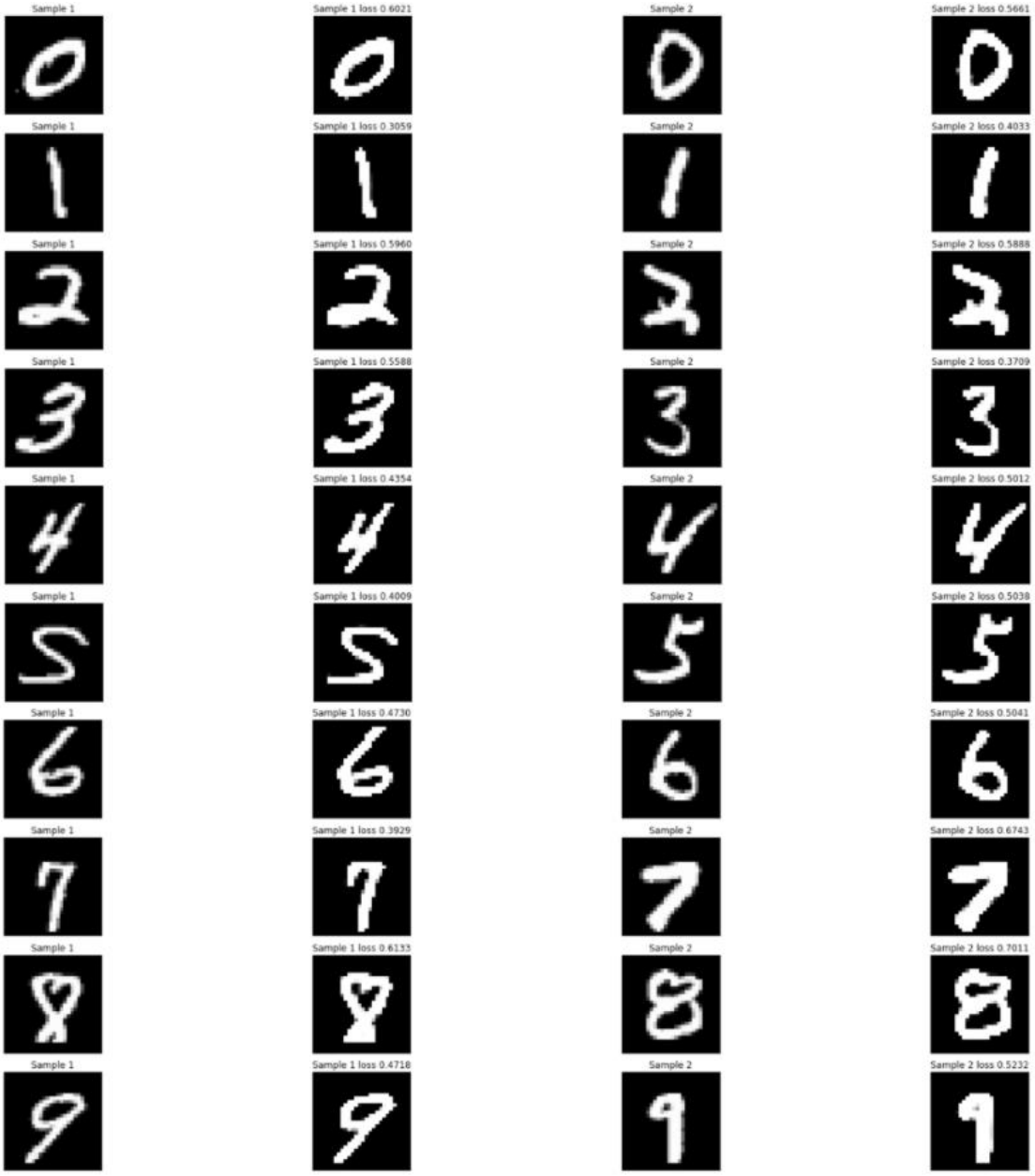
XVI. Here, for Training parameters used

- Relu activation function
- SGD is used as an optimizer
- learning rate of 1.0
- 10 epochs
- mini-batch size of 10
- MSEloss

Results:

Total parameters is = 74497
Encoder parameters is = 18816.0
Decoder parameters is = 55681.0
Average loss calculated for 10 epochs is = 0.47607351556420324

The 20 sample reconstructed images from testing data in the report (2 image for each class) along with the original images are:



GitHub Link:

https://github.com/SunilDevlops/Programs/blob/master/ComputerVision/ProgramAssignment2/Question2/MNIST_Dataset_Autoencoder_Conv.ipynb

Observations:

- FC mode takes more parameters as compare to the CONV mode
- FC mode produces less accurate results as compare to the CONV mode
- Average loss in FC mode is **0.4988**, whereas Average loss in CONV mode is **0.4761**
- Learning rate was initially taken as 0.1, but as there is very slow change in test_loss decrease pattern.
- And as we are checking only for 10 epochs, that is why learning rate is taken as 1.0
- Loss curve of CONV mode for both training and test set are consistently lower than loss curve of FC mode, which proves that CONV mode is better in this visualizing task as compare to FC mode.
- Below are the graphs: