In this task, we’re trying to train an auto encoder on the images obtained from the mean of images selected from cifar-10 and MNIST datasets.

1. Loading Datasets: First of all, we have to load datasets to be able to work with them. For this type of datasets with large size it’s logical to download them from torch and import them directly into our workspace. So, I imported datasets into two sets one for training and another for testing. I also performed necessary operations like resizing, extending channels and normalizing on the data while loading it using transformers.
2. Mean of images: In this part, I defined two functions one for getting the mean of the images of two datasets and another for converting images into tensors so later they can be used for training the auto encoder.

So I applied the ‘get\_mean’ function on both trainset and test sets of CIFAR and MNIST datasets. Then, I converted those mean images into tensors and made sure they have the right shape for any later use.

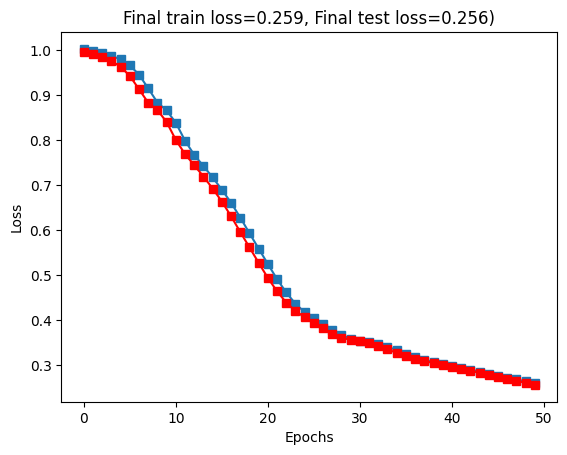
1. Creating the Auto encoder: In this section, I defined a class for auto encoder and then later in my notebook I created an instance of this class. Any auto encoder is made of and encoder and a decoder. First, we construct the encoder which in this case has three convolutional layers followed by leaky RELU.

There are many reasons behind the choice of leaky RELU. Some of the main ones are the possibility of faster convergence and more stable training that comes with this activation function. It also introduces non-linearity which is essential for auto encoder that aim to learn complex underlying structures of images.

The use of MSE for the loss function is because of it encouraging the model to learn pixel-wise similarities, that’s why it’s suitable for almost all of the image reconstruction tasks.

The Adam optimizer also accelerates convergence and it has an adaptive learning approach that adjusts the learning rate based on the gradients.

1. Training Loop: Now we have to somehow configure our training loop and set the number of epochs, size of data batches and so on. I set the number of epochs 50 and the batch size 32 and I used data loader for splitting data into batches for both train set and test set. The data loader is being brought through the forward and backward pass and training and testing losses are computed and stored in a list. So we can later plot the average training and testing loss for each of the epochs during the whole training process.
2. Visualization: Now we can see the overall result and changes in both of the losses through the whole process.



The End