EE5179: Deep Learning for Imaging

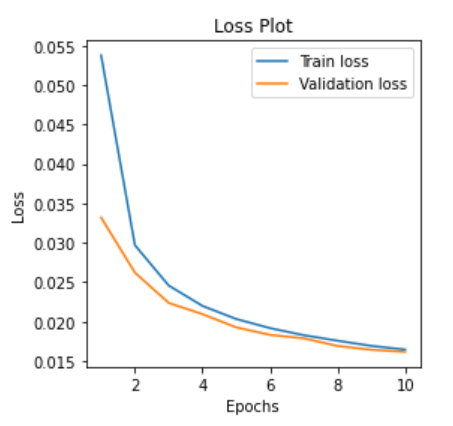
# Programming Assignment 4: Auto-encoders

## Comparing PCA and Autoencoders

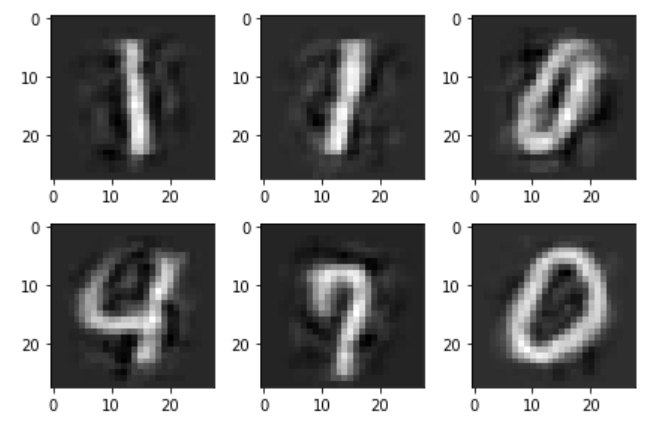
**Autoencoder design:**

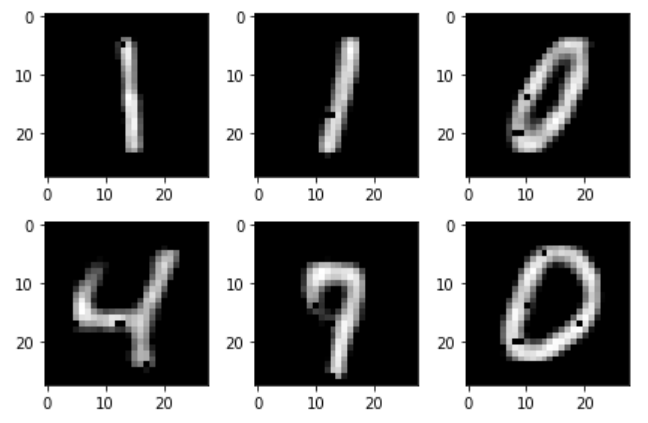
|  |  |
| --- | --- |
| **Encoder** | **Decoder** |
| – input (784) | – fc (128) |
| – fc (512) | – fc (256) |
| – fc (256) | – fc (784) |
| – fc (128) |  |
| – fc (30) |  |

**Training plot for the autoencoder:**



*Test loss: 0.017%*

**Visual Comparison of reconstruction using AE and PCA (number of components = 30):**



Reconstruction using PCA Reconstruction using AE

**Observations:**

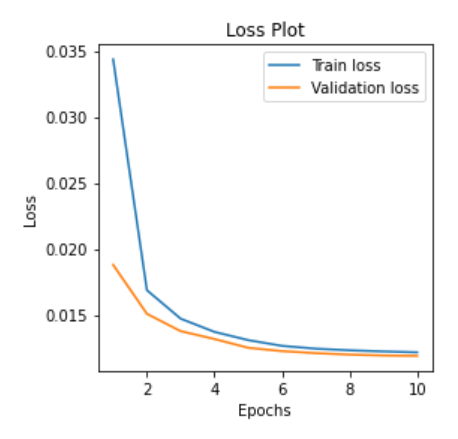
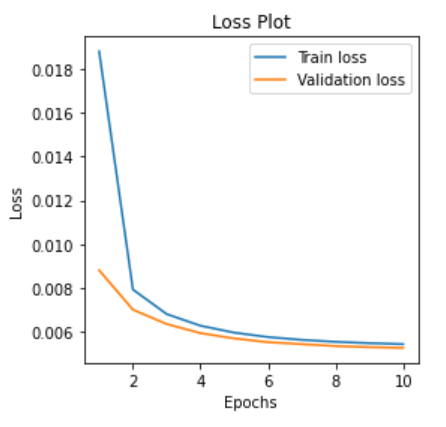
* Using a greater number of components in PCA leads to better reconstruction
* Using an autoencoder has a much better reconstruction in contrast to PCA

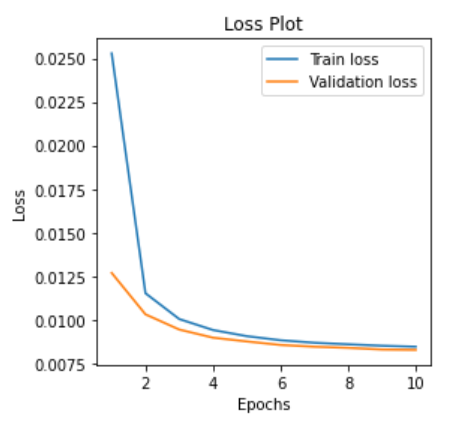
## Experimenting with hidden units of varying sizes

**Autoencoder design:**

The architecture consists of only a hidden layer and the output layer: fc(x)-fc(784)

X is varied from 64, 128, 256 to perform this experiment.

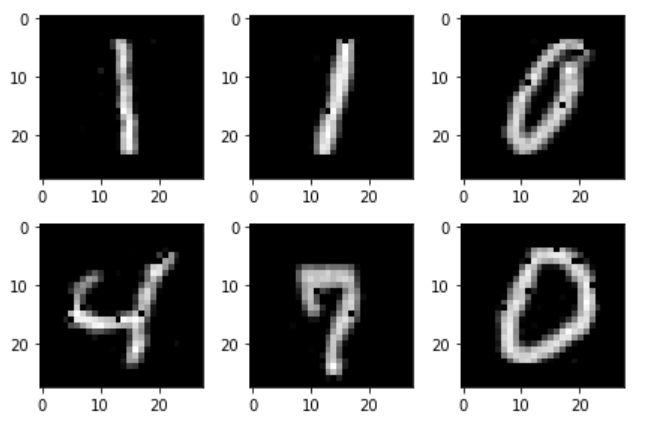
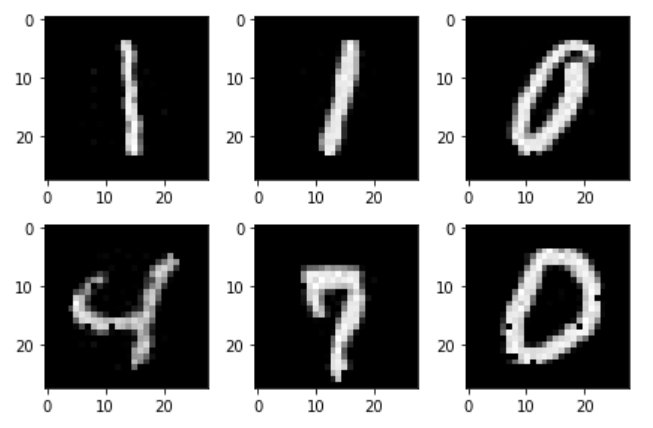
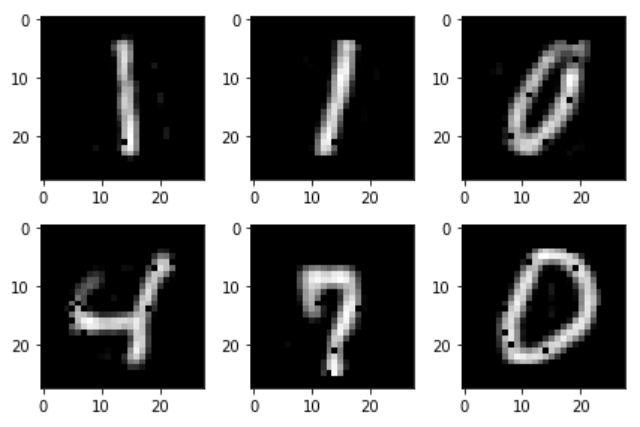
**Training loss plots:**



Hidden size = 64 Hidden size = 128 Hidden size = 256

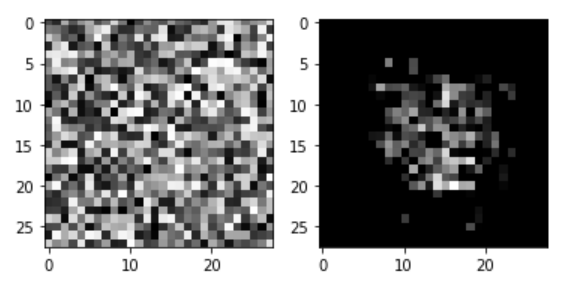
**Test losses:**

|  |  |
| --- | --- |
| Hidden size | Test loss |
| 64 | 0.013% |
| 128 | 0.009% |
| 256 | 0.006% |

**Visual comparison of reconstruction:**

Hidden size = 64 Hidden size = 128 Hidden size = 256

**Reconstruction of noise image:**



Input: noise image Output: reconstructed

image (hidden layer size = 256)

**Observations:**

* Increasing the hidden layer size improves the reconstruction
* There are a few black pixels noticeable in the reconstruction. The number of these reduce with increase in hidden layer size.
* Reconstruction of a noise image outputs an image that has a lot of the pixels towards the corners removed. This shows that the encoder learns the digits and removes the background pixels.

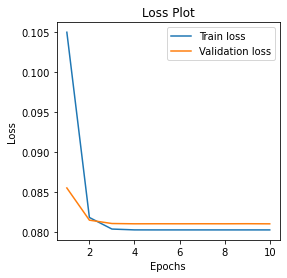
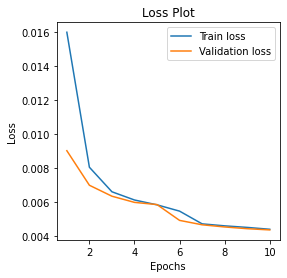
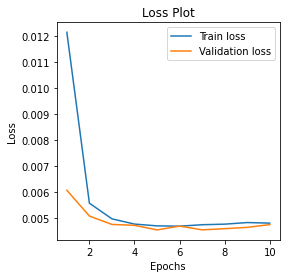
## Sparse Autoencoders

**Autoencoder design:**

Very similar to the autoencoder in the previous case, we have an AE with a single hidden layer. This will be an autoencoder that is overcomplete. We take hidden layer size = 1225. fc(1225)-fc(784)

**Training loss plots:**

Sparse autoencoders are regularised autoencoders. They use L1 penalty. We increase sparsity by increasing the penalty on the regularisation term, LL1.

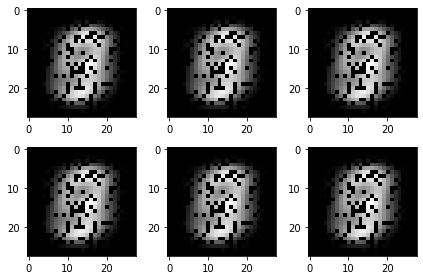
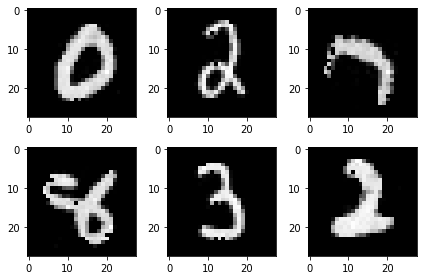
  

LL1 = 1e-4 LL1 = 1e-7 LL1 = 1e-10

**Test losses:**

|  |  |
| --- | --- |
| LL1 | Test loss |
| 1e-4 | 0.080% |
| 1e-7 | 0.004% |
| 1e-10 | 0.005% |

**Visual comparison of reconstruction:**

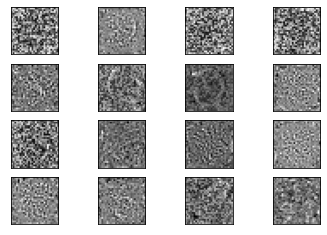
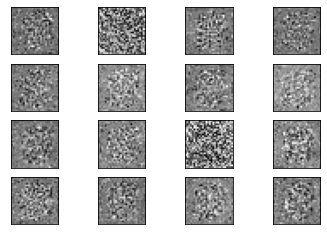
LL1 = 1e-4 LL1 = 1e-7 LL1 = 1e-10

**Average activation:**

Average activations compared between sparse autoencoder with different activations and the standard AE (hidden layer size = 256).

|  |  |
| --- | --- |
| AE | Average activation |
| SAE, LL1 = 1e-4 | ≈ 0 |
| SAE, LL1 = 1e-7 | 0.222 |
| SAE, LL1 = 1e-10 | 1.556 |
| AE | 0.940 |

**Decoder weights:**

SAE (LL1 = 1e-7) Standard AE

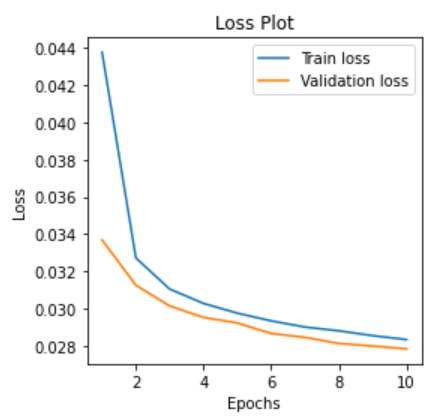
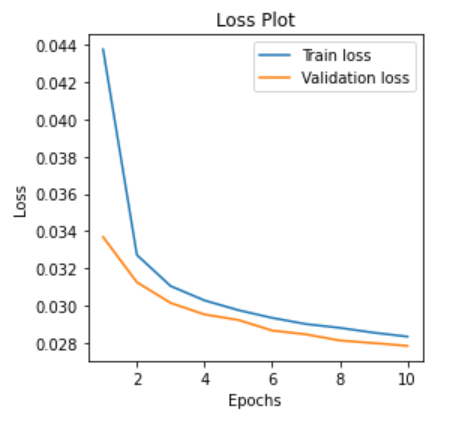
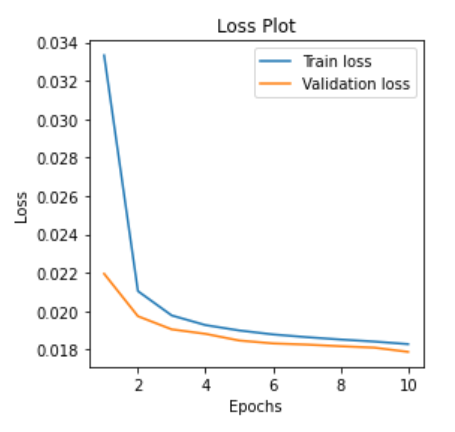
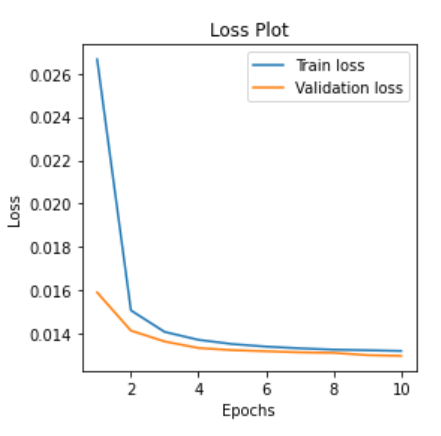
**Observations:**

* Very high sparsity results in very poor reconstruction.
* Higher the sparsity, lower the average activation.
* At LL1 = 1e-7, the SAE performs the best. We compare this model with a standard AE of hidden layer size 256.
* In comparison to 1225 hidden layer size in SAE, and Standard AE of hidden layer size 256, the two have comparable total activation per neuron.
* The compared decoder weights show that the SAE learns the structure of the outline of the digits better. The AE on the other hand has learned weights that resemble noise.

## Denoising Autoencoders

**Autoencoder design:**

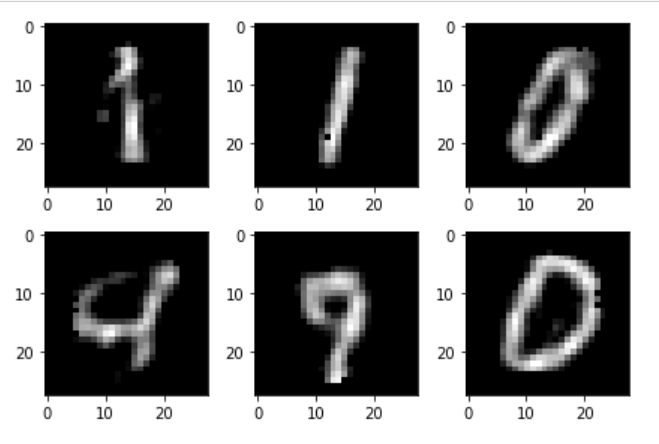
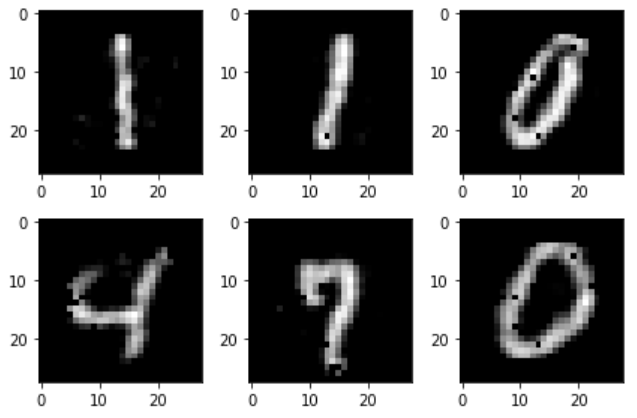
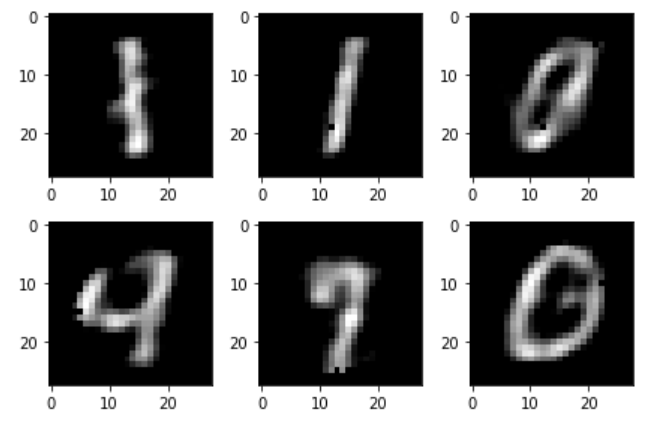
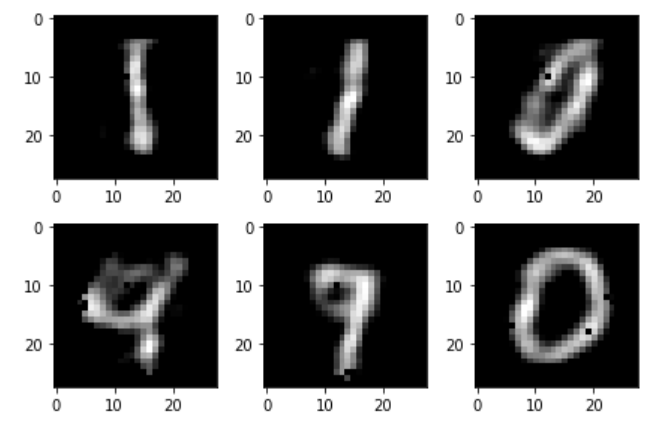
This DAE has a single hidden layer of size 256. Regularisation done by varying the noise factors 0.3, 0.5, 0.8, 0.9.

**Training loss plots:**

Noise factor = 0.3 Noise factor = 0.5 Noise factor = 0.8 Noise factor = 0.9

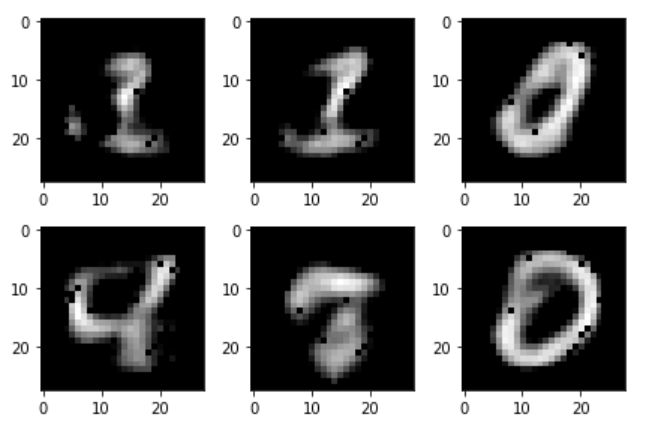
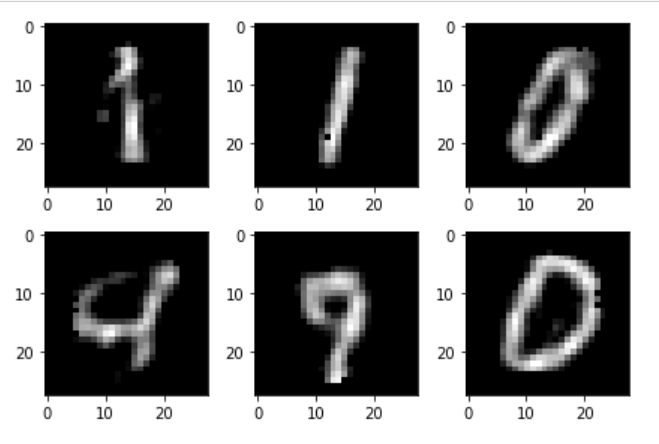
**Test losses:**

|  |  |
| --- | --- |
| Noise factor | Test loss |
| 0.3 | 0.014% |
| 0.5 | 0.019% |
| 0.8 | 0.029% |
| 0.9 | 0.032% |

**Visual comparison of reconstruction:**

NF = 0.3 NF = 0.5 NF = 0.8 NF = 0.9

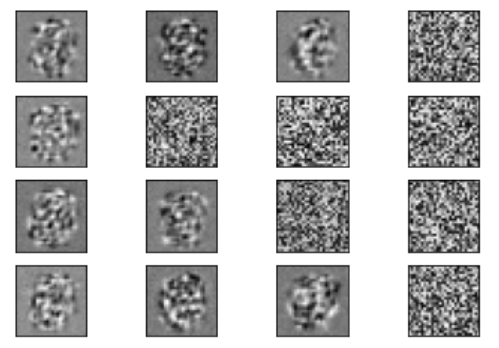
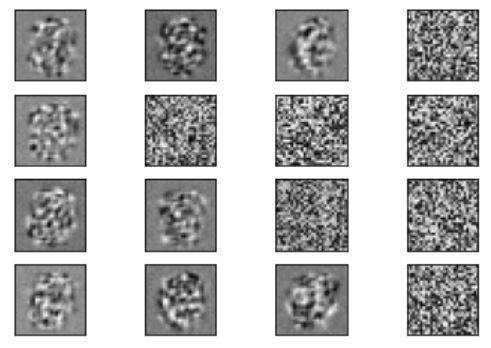
**Passing noisy images to Standard AE and DAE: Comparison:**

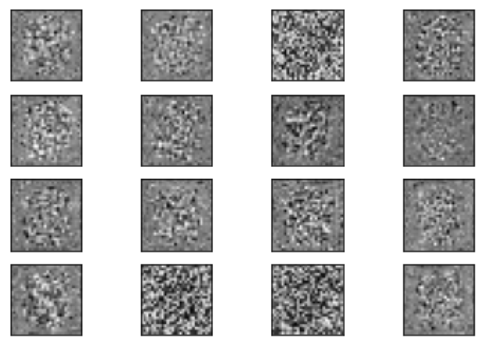
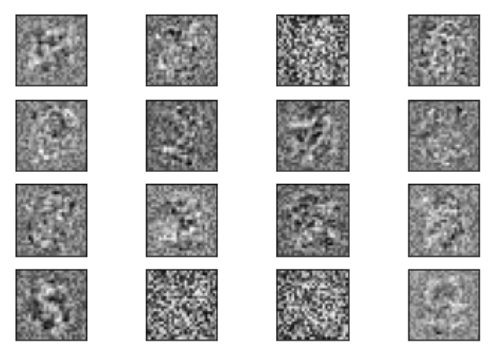


Output from Standard AE Output from DAE

when noisy input is passed

**Learned filters:**



Learned encoder and decoder weights of DAE

Learned encoder and decoder weights of standard AE

**Observations:**

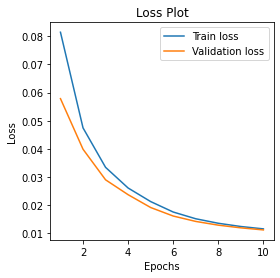
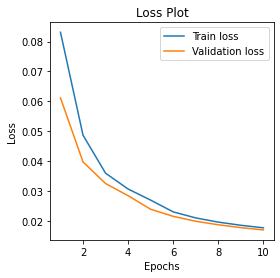
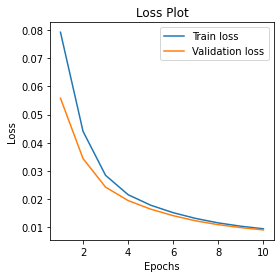
* Passing noise corrupted images through a standard AE, results in poor reconstruction. We see how DAE effectively learns the underlying distribution of the image data independent of the noise.
* Increasing the noise factor results in poorer test accuracy. However, visually we see that the model reconstructs better when the noise factor is increased to about 0.8.
* Upon observing the learned weights of DAE and standard AE, we see how the DAE weights better capture the variations.

## Convolutional Autoencoders

**Autoencoder design:**

|  |  |  |  |
| --- | --- | --- | --- |
| Encoder | Decoder: Unpool | Decoder: Deconvolve | Decoder: Unpool + deconvolve |
| Input-Conv1(8 3x3 filters with stride 1) | Unpool (7x7x16 to 14x14x16) | Conv1 (7x7x16 to 9x9x16) | Unpool |
| 2x2 Maxpooling | Conv2 (14x14x16 to 14x14x8) | Conv2 (10x10x16 to 14x14x8) | Conv1 |
| Conv2(16 3x3 filters with stride 1) | Unpool (14x14x8 to 28x28x8) | Conv3 (14x14x8 to 28x28x1) | Unpool |
| 2x2 Maxpooling | Conv3 (28x28x8 to 28x28x1) |  | Conv2 |
| Conv3(16 3x3 filters with stride 1) |  |  | Unpool |
| 2x2 Maxpooling |  |  | Conv3 |

**Training loss plots:**

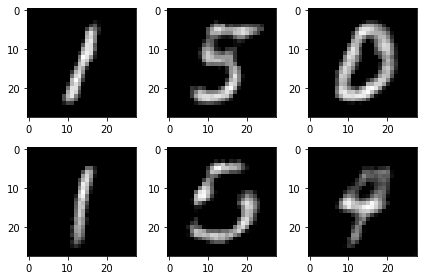
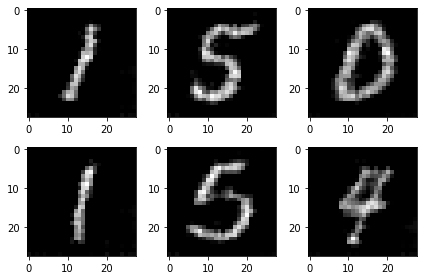
  

Unpool Deconv Unpool+Deconv

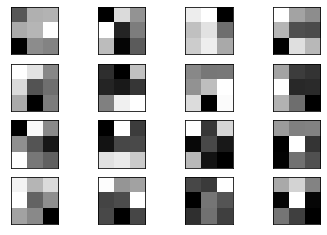
**Test losses:**

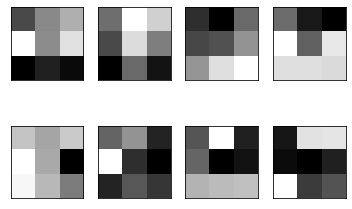
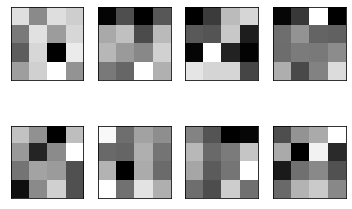
|  |  |
| --- | --- |
| Decoder model | Test loss |
| Unpool | 0.012% |
| Deconv | 0.017% |
| Unpool+Deconv | 0.009% |

**Reconstruction visualisation:**

Unpool Deconv Unpool+Deconv

**Decoder weights:**



Unpool Deconv Unpool+Deconv

**Observations:**

* The model is trained at a lower learning rate (=0.0001)
* The reconstruction is better with unpooling. The decoder weights also show the learning of the edges
* Upsampling using unpool+deconv has kernel weights that don’t learn edges or weights very well