Programming assignment 4: Autoencoder

Arunima Sarkar

1 Comparing PCA and Autoencoders

The MNIST image, upon flattening is 784×1 vector, and if we do a PCA on it with the largest 30 eigenvectors and reconstruct it then the error is 0.01753543473942351.
I also build a vanilla autoencoder and reconstructed the images from that, below is the loss curve.

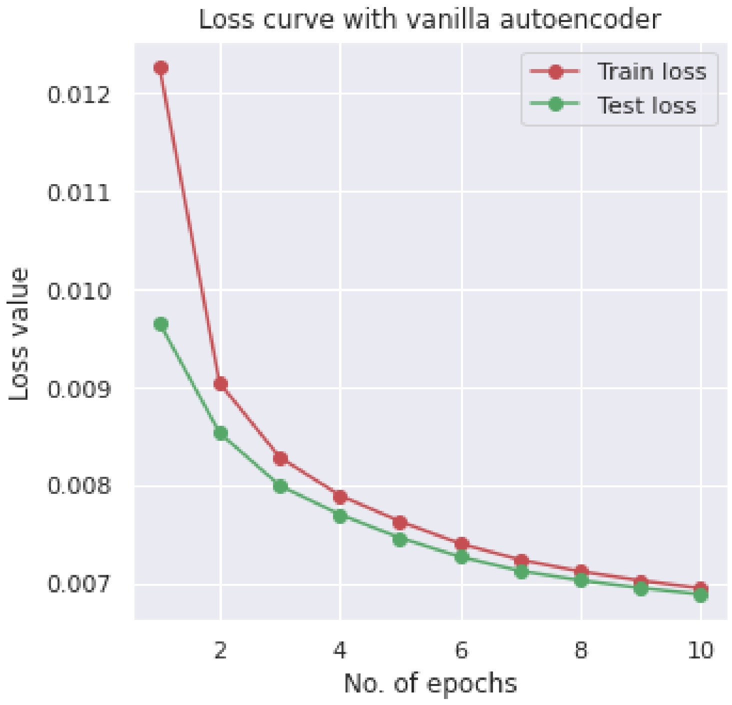


Figure 1: Vanilla Autoencoder

Comparing Visually the PCA and Autoencoder Output

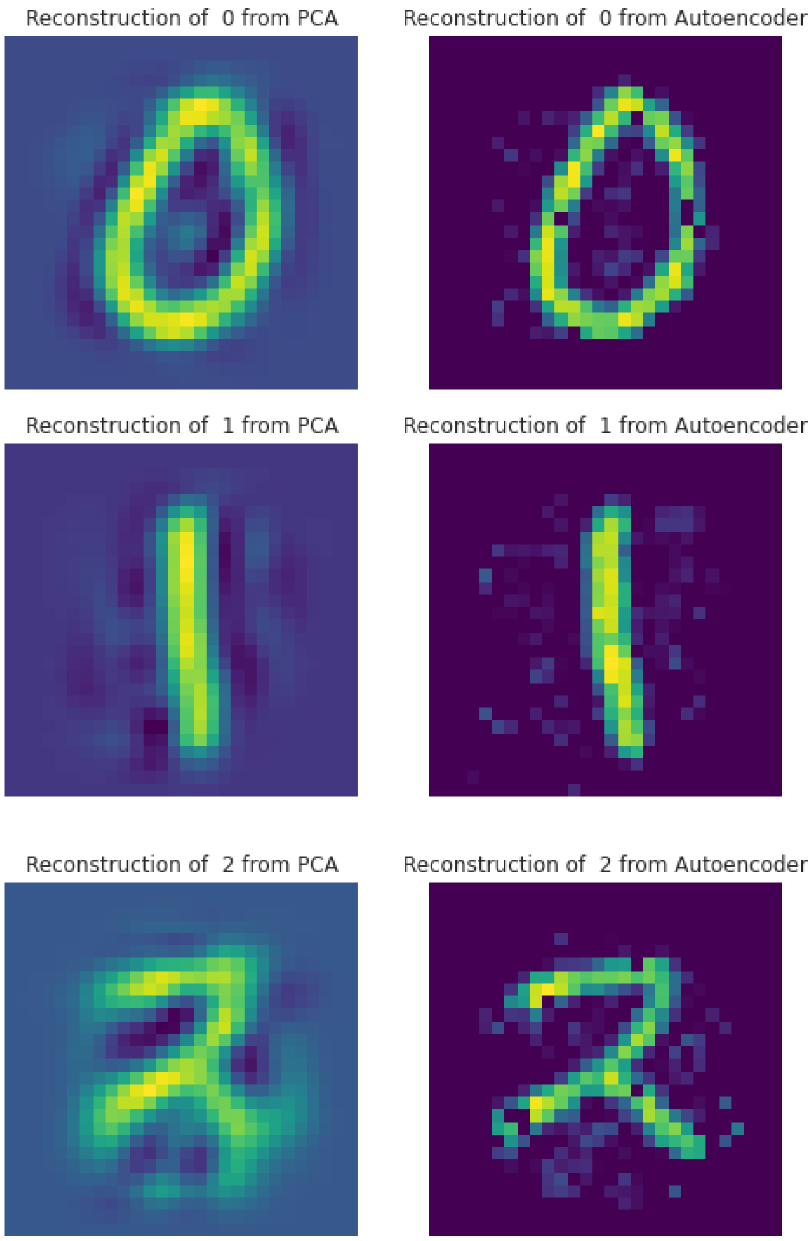


Figure 2: PCA and autoencoder Outputs for 0,1,2

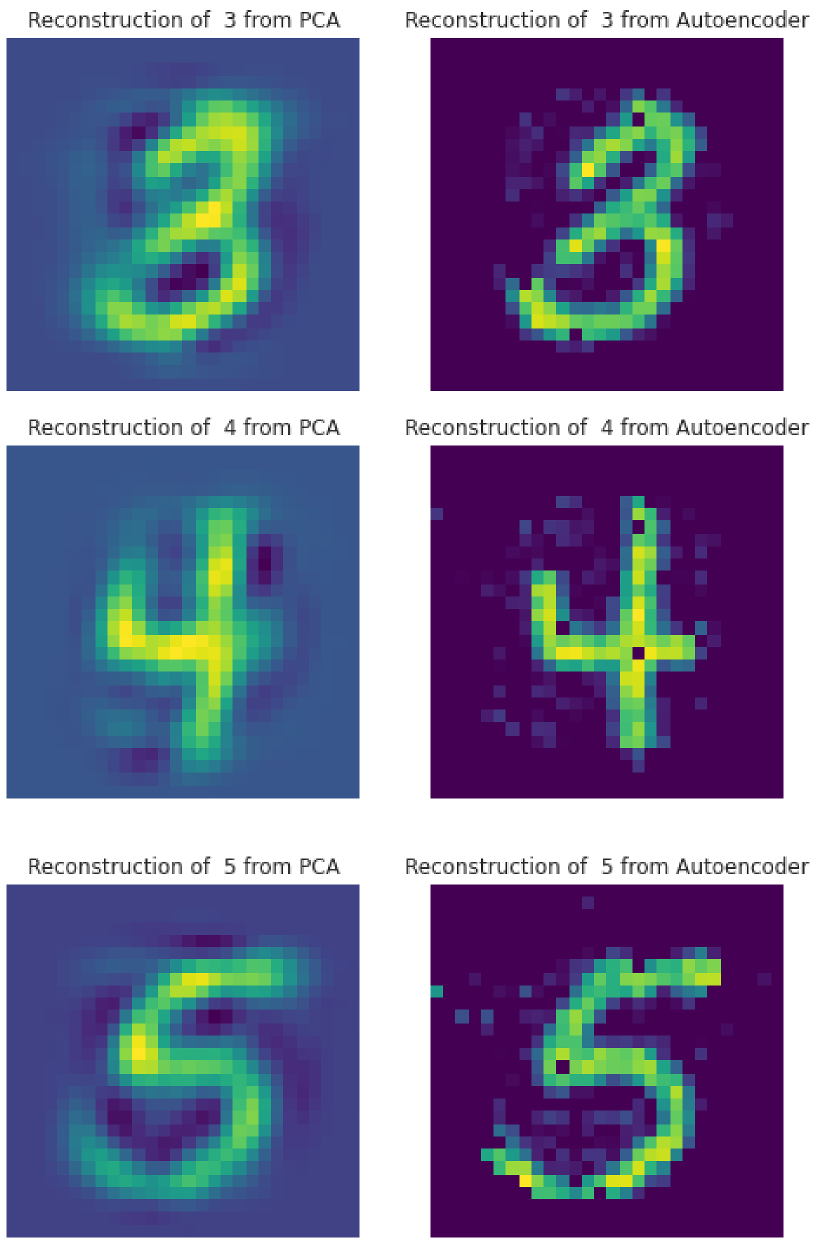


Figure 3: PCA and autoencoder Outputs for 3,4,5

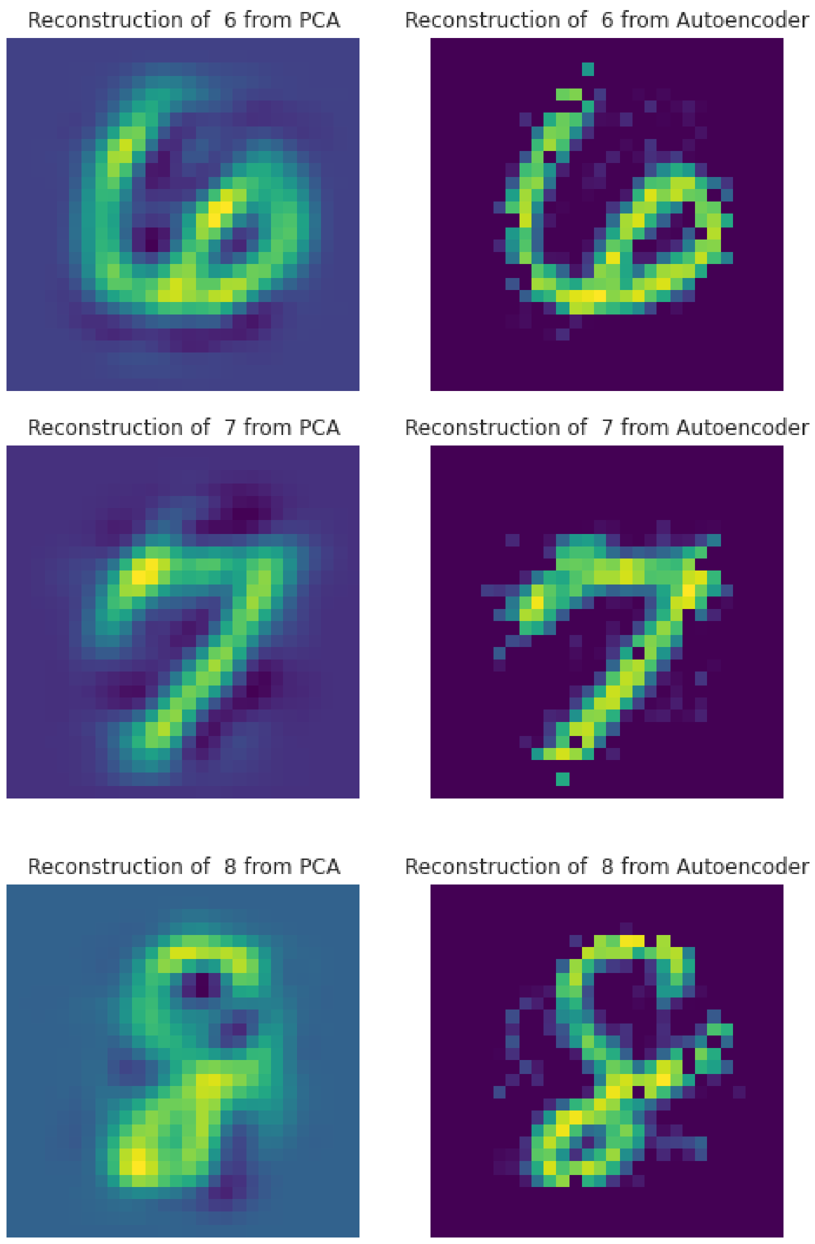
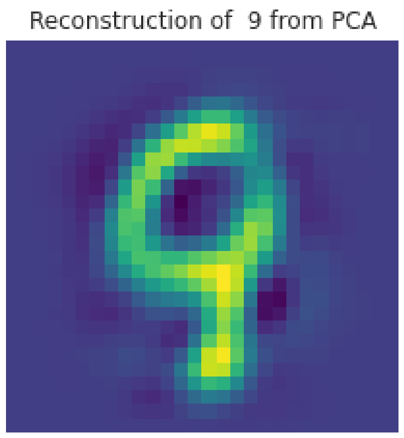
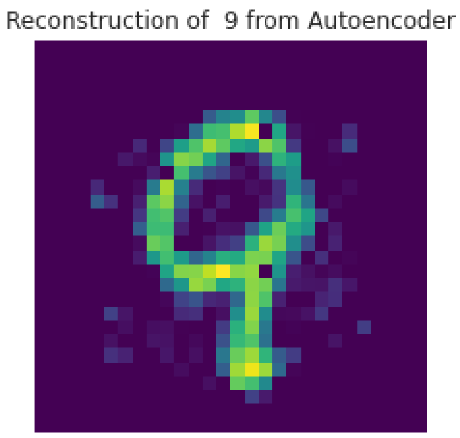


Figure 4: PCA and autoencoder Outputs for 6,7,8

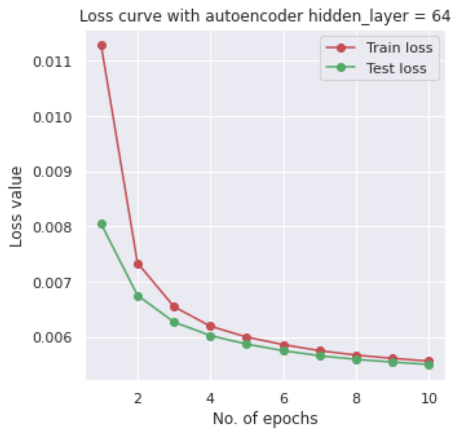




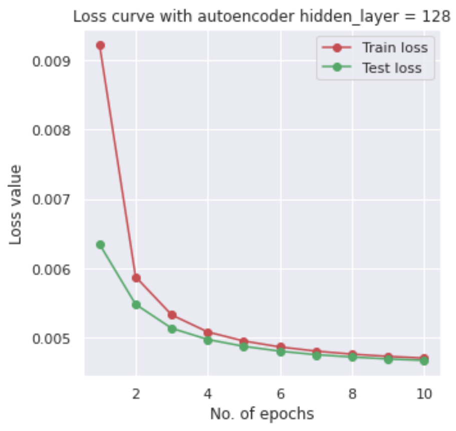
We can see that with Autoencoders the loss value falls to 0.007 with 10 epochs on test dataset with is much lower compared to the reconstruction accuracy from PCA. Even Visually its evident that PCA reconstructed images are more blurry whereas Autoencoder images have sharper edges.

2 Experimenting with hidden units of varying sizes

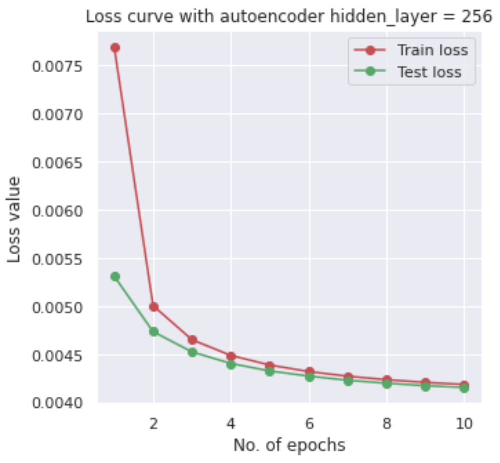
2.1 Training plots for x $= [64, 128, 256]$



((a))



((b))

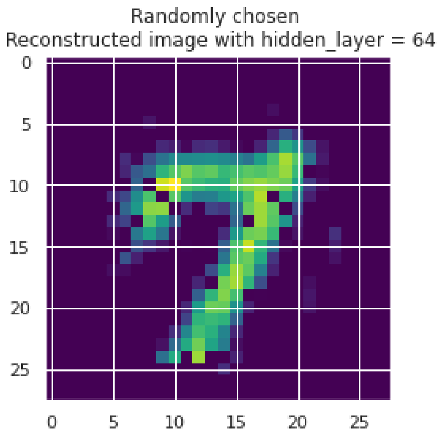


((c))

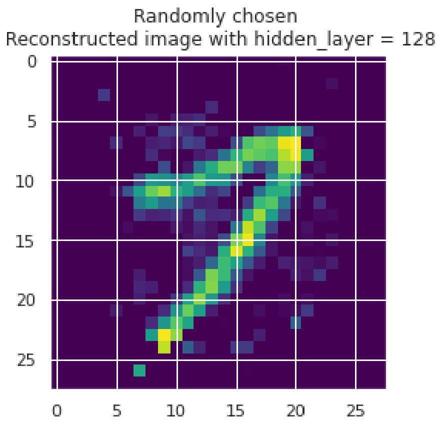
Figure 6: Loss curves

2.2 Randomly chosen image being reconstructed

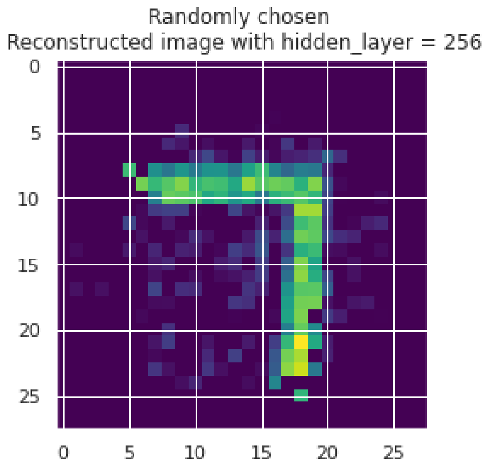
I have experimented with digit 7 with x = $[64, 128, 256]$ and below are the results of reconstruction.Its evident that as the number of layers increases the reconstruction gets better.



((a))



((b))



((c))

Figure 7: Reconstruction of 7(randomly chosen)

2.3 Non-digit image and noise being reconstructed

I used the mandil.png image as a non-digit image along with gaussian noise, and below are the results.

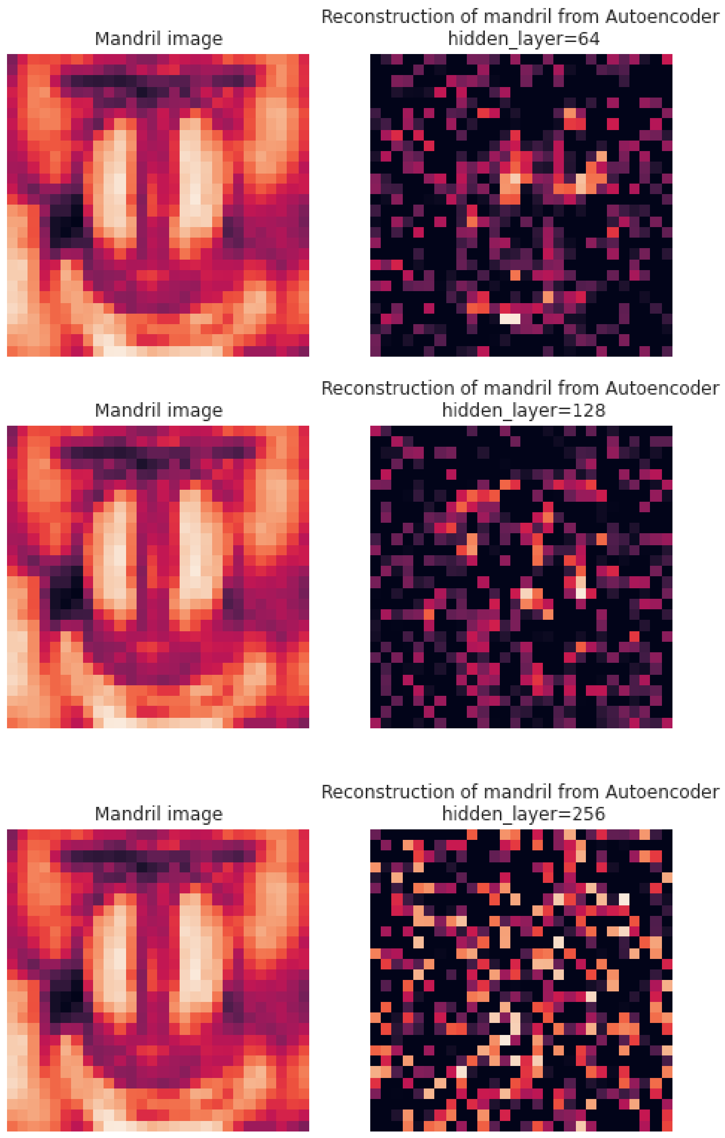


Figure 8: Reconstructed image for Mandril

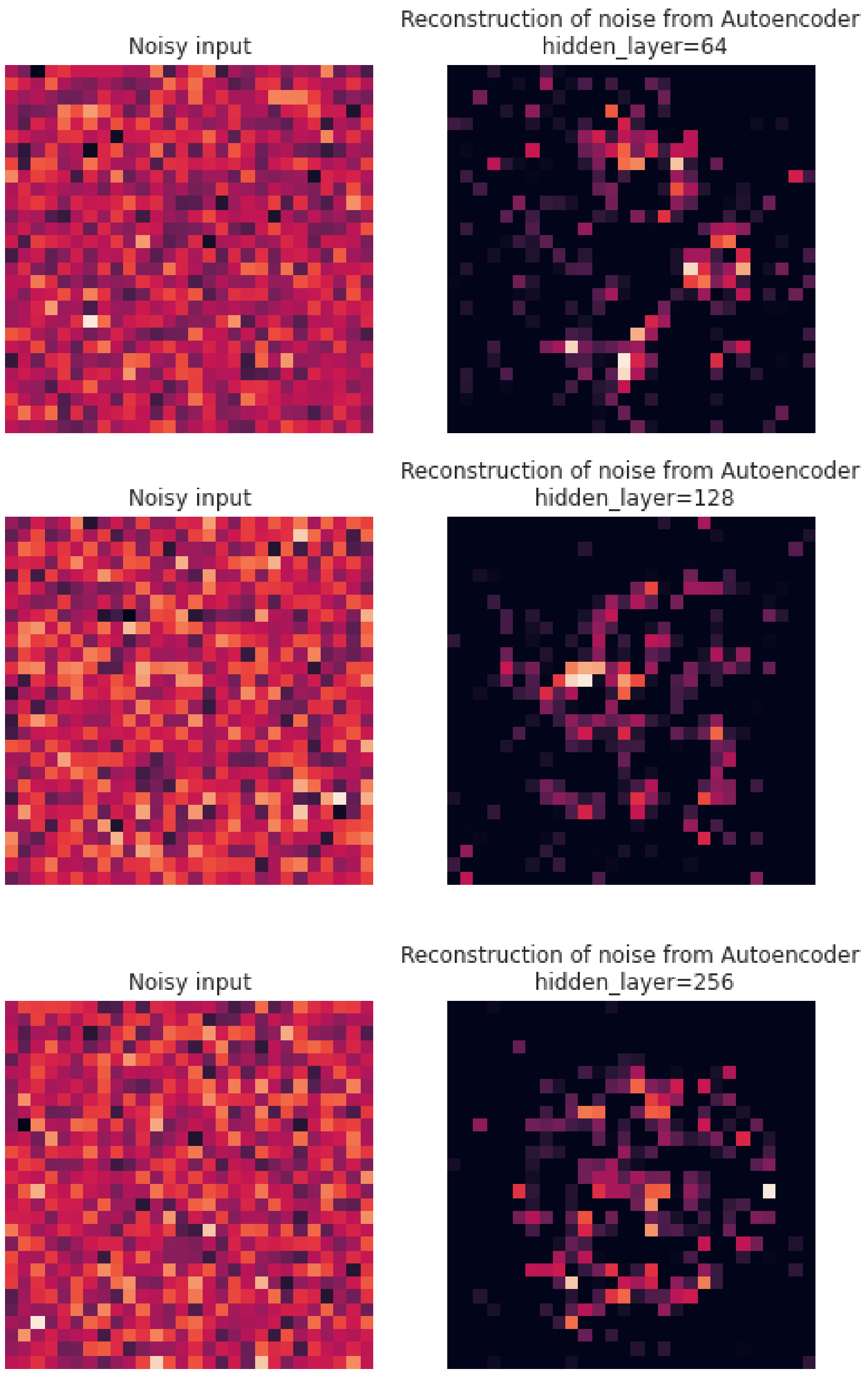


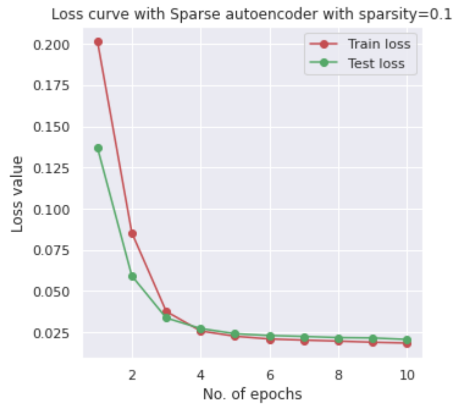
Figure 9: Reconstructed image for standard gaussian

Inferences drawn

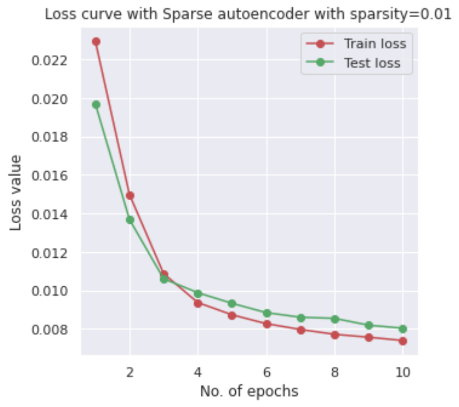
- If we pass a non-digit image or Noise we get a meaningless output. As noise was Out-of-Distribution and same is seen from the mandril image.
- Higher the hidden layer more accurate will be the reconstruction.

3 Sparse Autoencoder

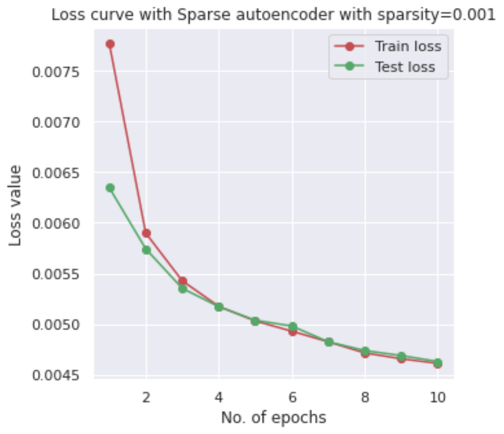
3.1 Loss plots for different sparsity



((a))



((b))

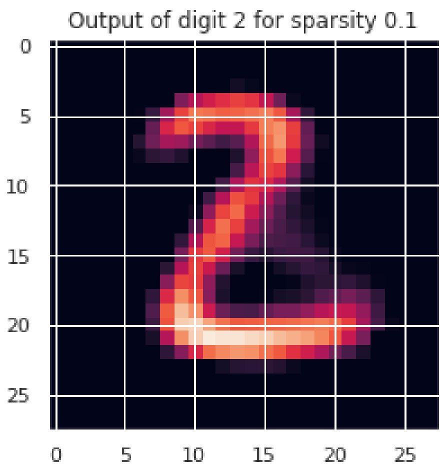


((c))

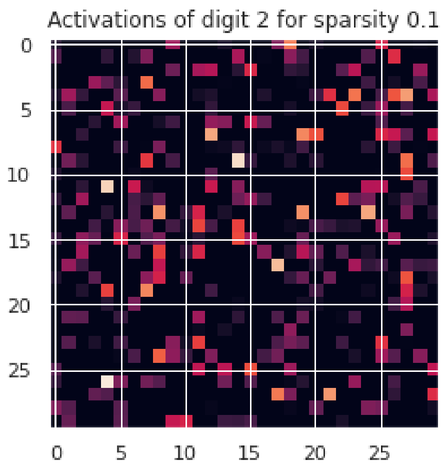
Figure 10: Loss curves

3.2 Average hidden layer activation and related sparsity

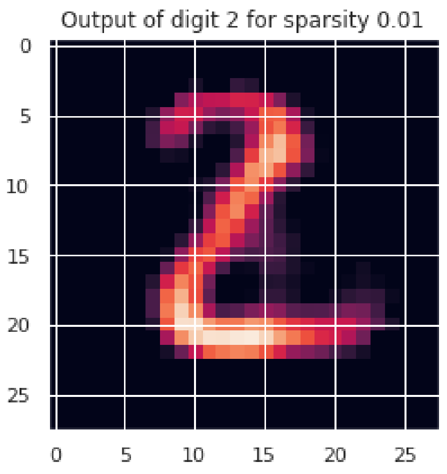
I choose 2 as the digit for checking sparsities. The plots for different sparsities are.



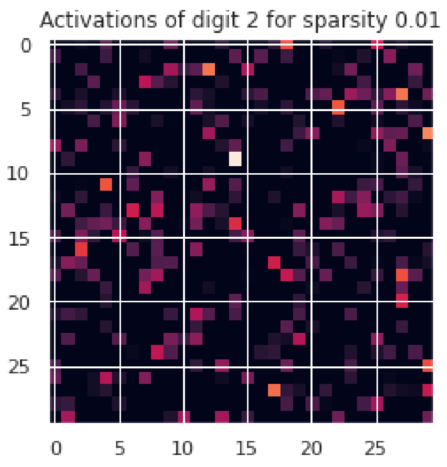
((a)) Output



((b)) Encoded output

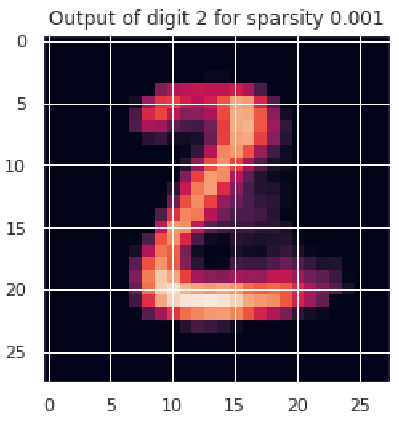


((c)) Output

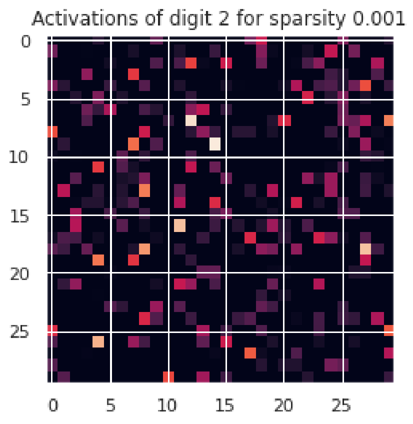


((d)) Encoded output

Figure 11: Sparsity plots



((a)) output



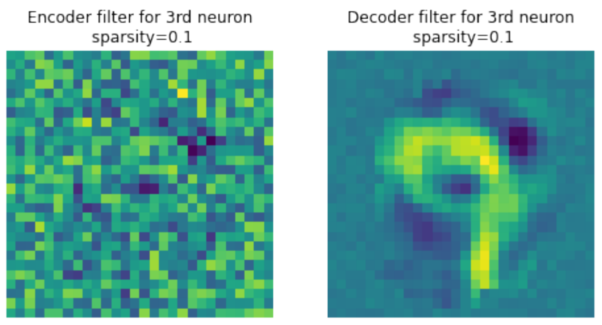
((b)) Encoder output

Figure 12: Sparsity plots and output

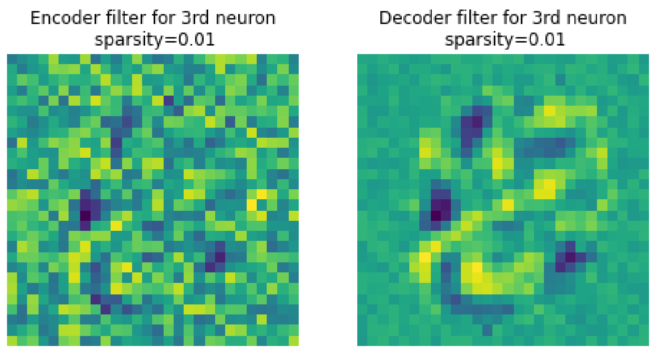
The average activations for Standard encoder with hidden layer 64 is 0.13396140904426573, with 128 it is 0.10157207226753234, and with 256 it is 0.07290675175189971. So it is decreasing with increasing number of hidden layers. Whereas, the average activation for Sparse Autoencoder with sparsity of 0.1 is 0.00013127546962350608, for 0.01 is 0.0009793924264609813 and for 0.001 is 0.005324395973980427.

3.3 Encoder-Decoder filter

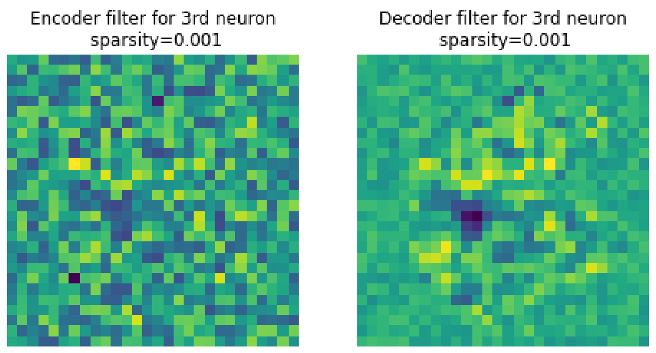
The encoder-decoder filters for 3rd neuron of Sparse Autoencoder with varying sparsity is shown below.



((a))



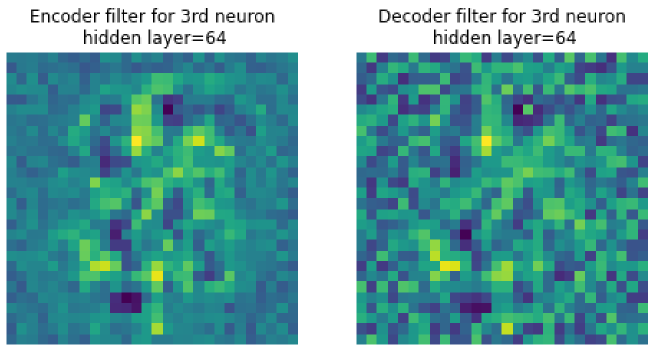
((b))



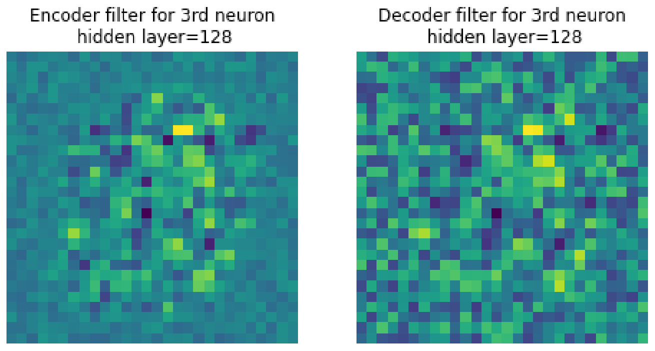
((c))

Figure 13: Sparsity plots

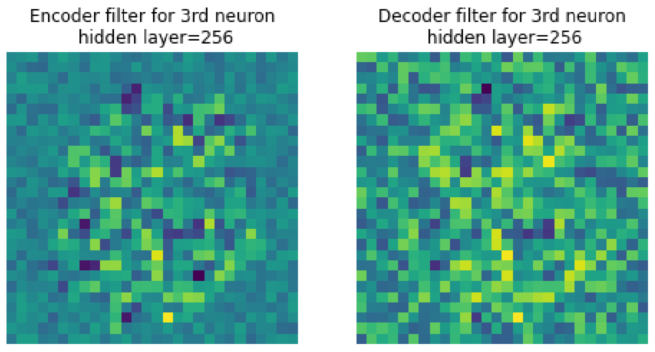
The encoder-decoder filter for Standard autoencoder is,



((a))



((b))



((c))

Figure 14: Sparsity plots

Inferences from Sparse Autoencoder

- The test and train losses both hits a higher values as the sparsity parameter decreases, which is evident as it generalizes the model and prevents from over-fitting.
- We can observe that as regularization increases the average activations decreases.The regularizations makes the neuron fire less, which satisfies the functionality of Sparseness.For standard autoencoder the average activations keeps on decreasing which make the neurons more susceptible to firing.
- The decoder if not the encoder filter resembles a certain shape in sparse autoencoder, which indicates that they reconstruct output as digits better. Whereas in standard autoencoder filters, the encoder or decoder filters doesn't make much sense.

4 Denoising Autoencoder

4.1 Loss plots compared to Standard Autoencoder

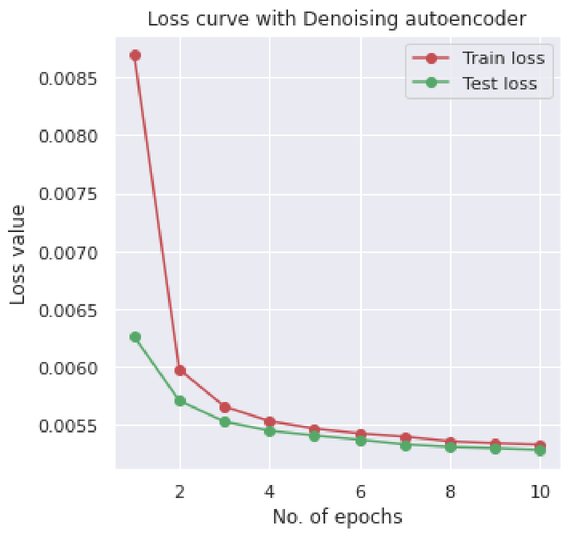


Figure 15: Corrupted with Gaussian Noise

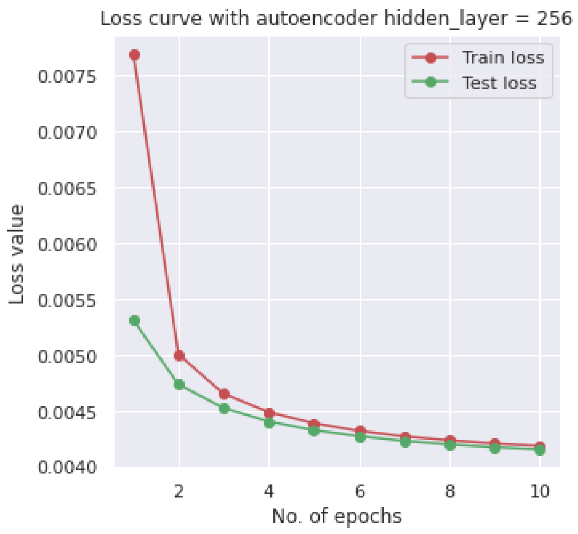
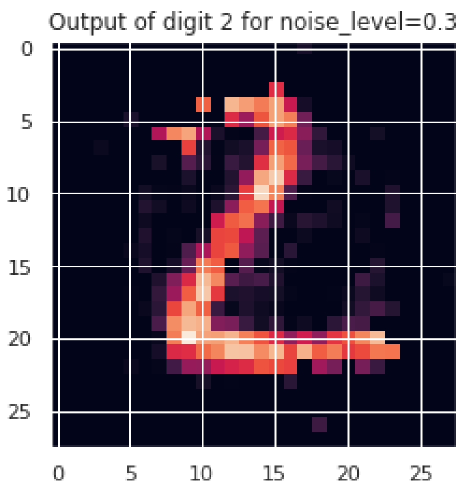
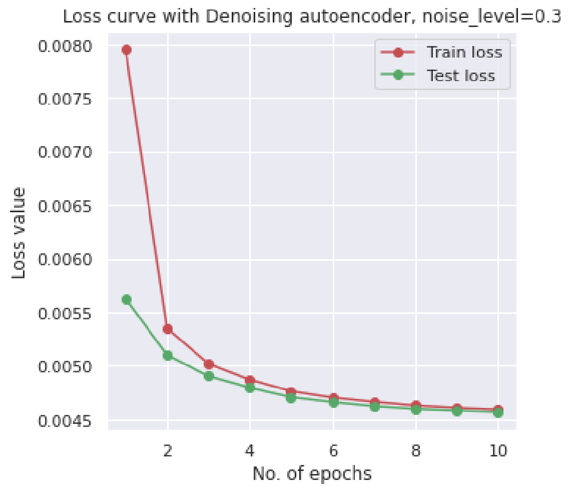


Figure 16: Standard Autoencoder

4.2 Varying Noise level

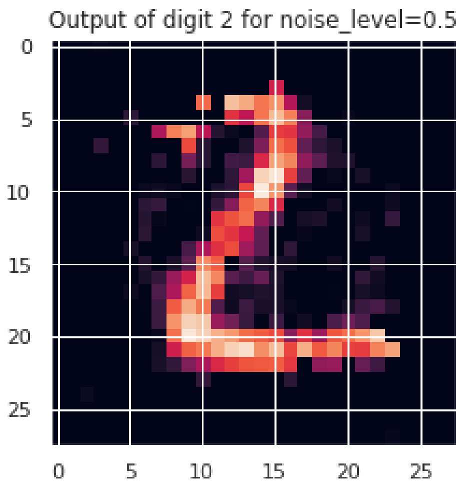


((a)) Sample reconstructed output

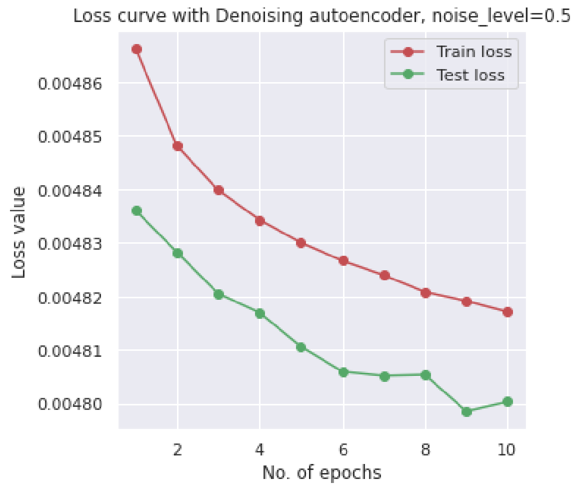


((b)) Loss curve

Figure 17: Noise level 0.3

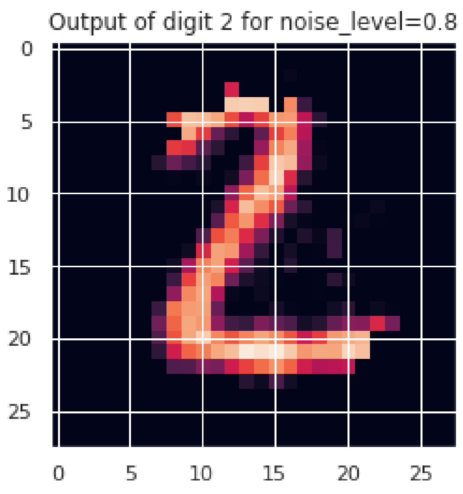


((a)) Sample reconstructed output

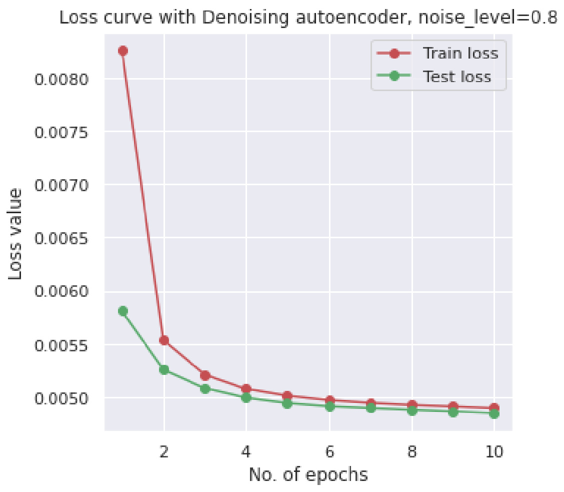


((b)) Loss curve

Figure 18: Noise level 0.5

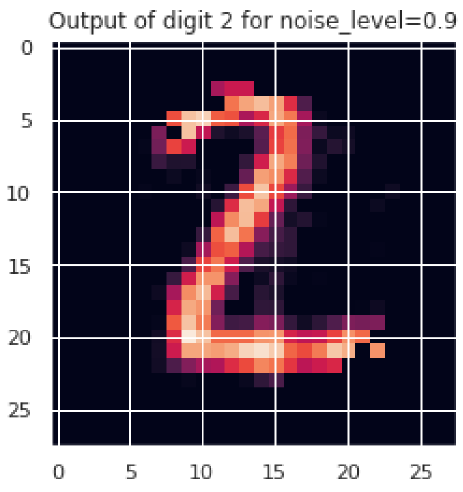


((a)) Sample reconstructed output

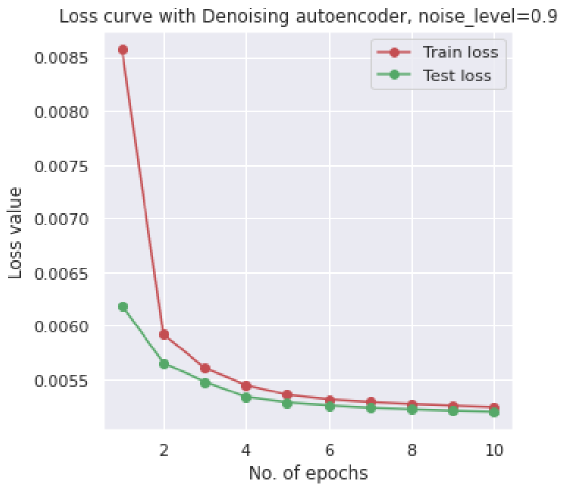


((b)) Loss curve

Figure 19: Noise level 0.8



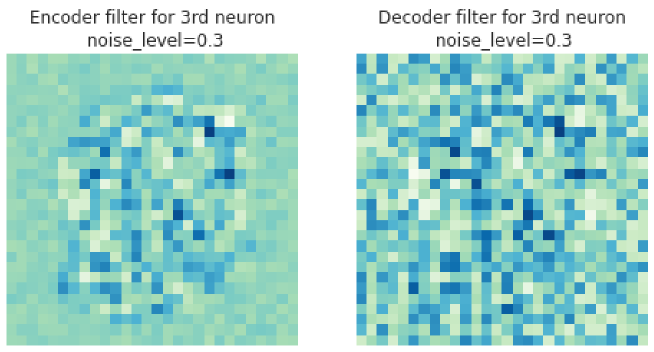
((a)) Sample reconstructed output



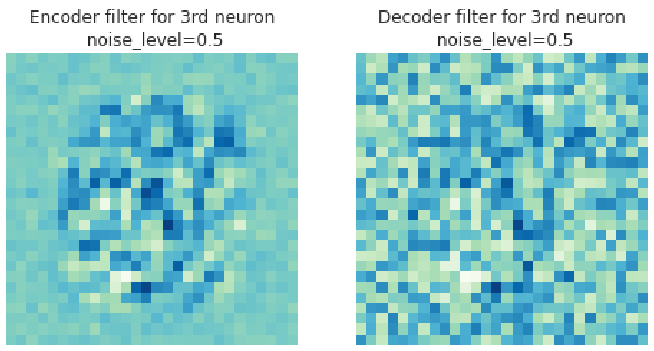
((b)) Loss curve

Figure 20: Noise level 0.9

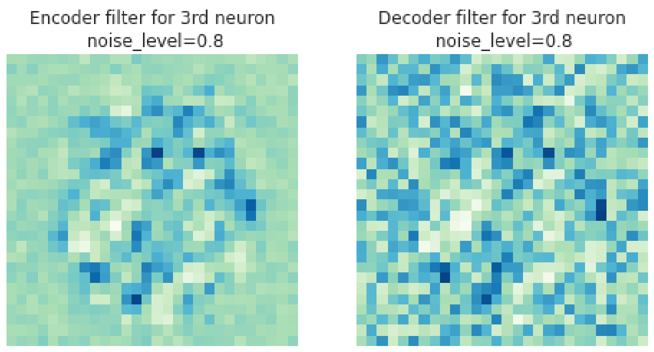
4.3 Learned filters of Denoising Autoencoder



((a))

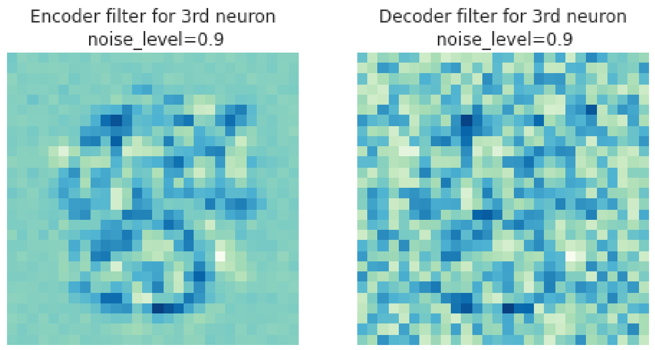


((b))

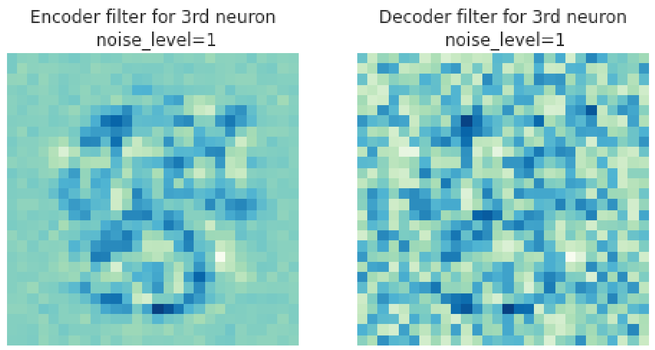


((c))

Figure 21: Encoder-Decoder plots



((a))



((b))

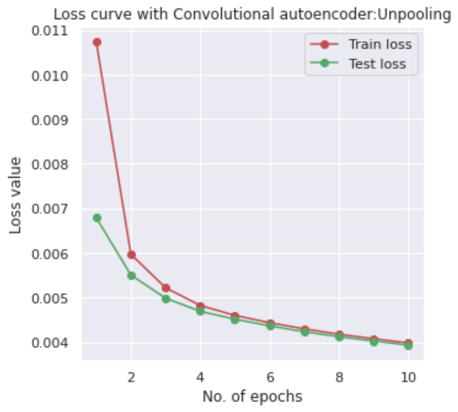
Figure 22: Encoder-Decoder plots

Inferences drawn

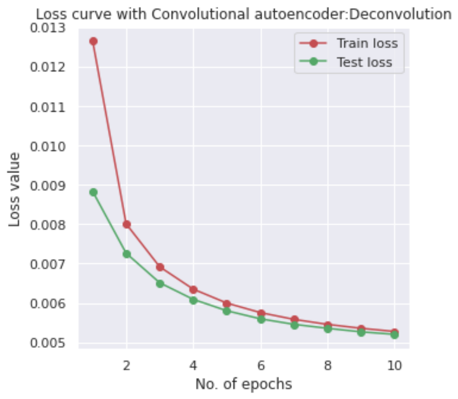
- From Figure 15 and Figure 16 it is evident that using noise doesn't improve reconstruction.
- The reconstruction error might be high but still I received a visually complete picture of the input with higher noise.
- Higher noise level gives a better encoder picture compared to standard autoencoder.

5 Convolutional Autoencoder

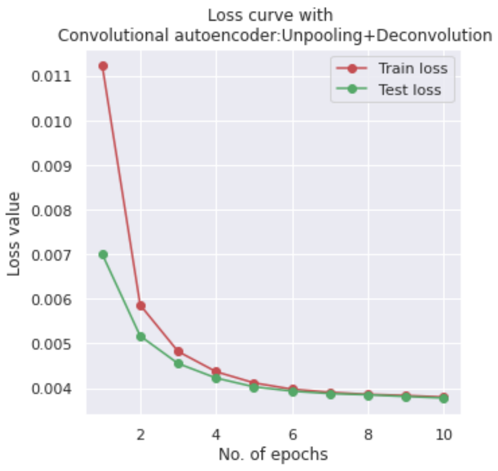
5.1 Plotting the losses and reconstructed output



((a))

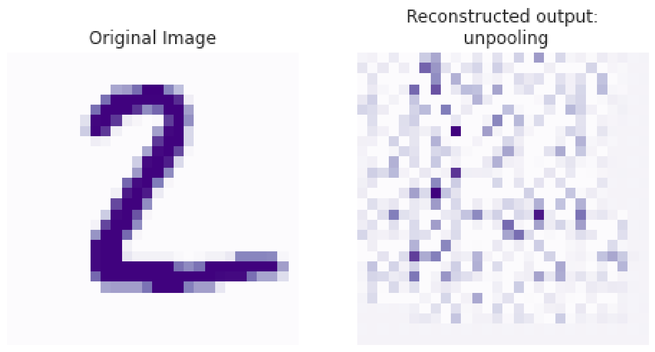


((b))

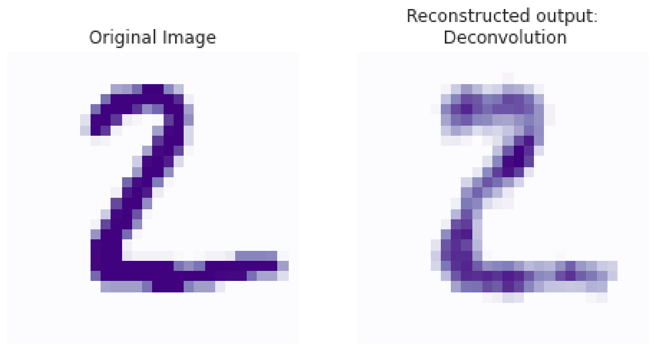


((c))

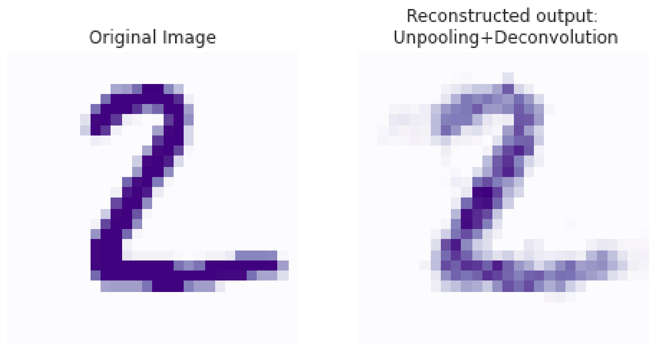
Figure 23: Loss curves



((a))



((b))



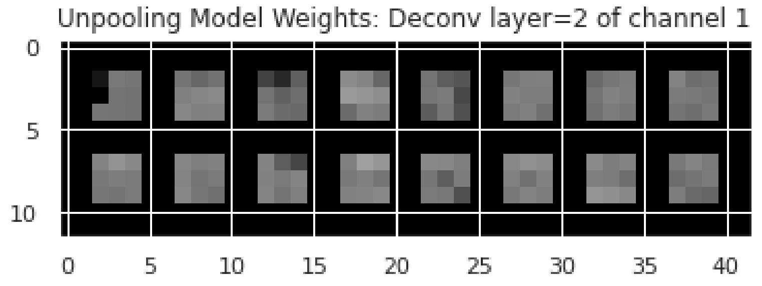
((c))

Figure 24: Reconstruction plots

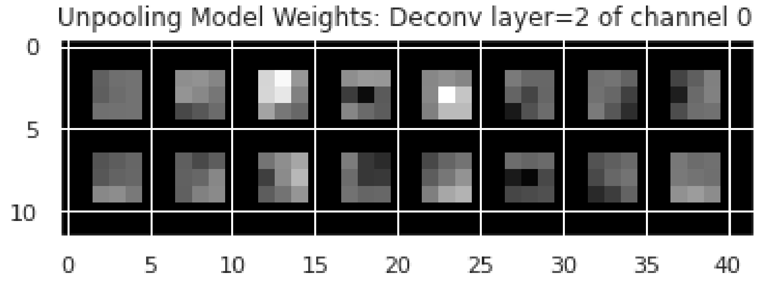
5.2 Filters for Unpooling Model

Here I choose to show only output of 3 channel.

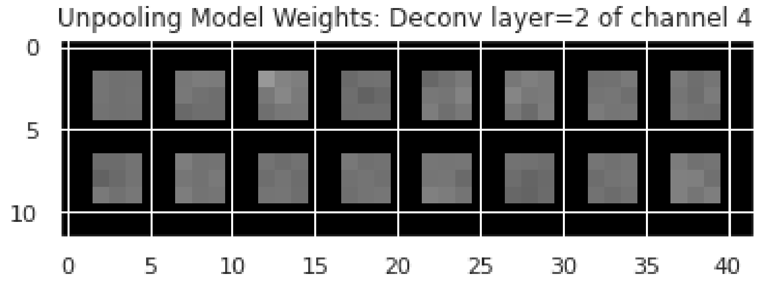
Conv2



((a))



((b))



((c))

Figure 25: Decoder plots

Conv3

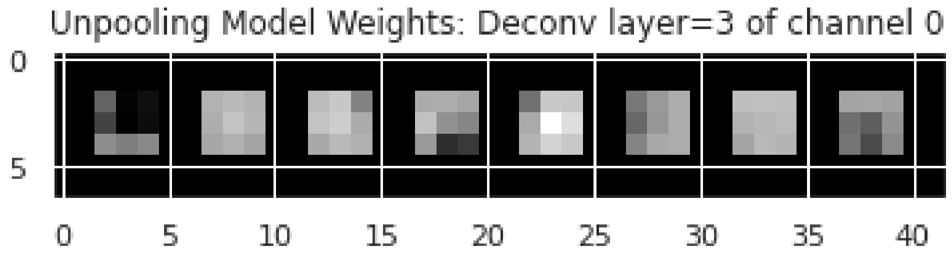
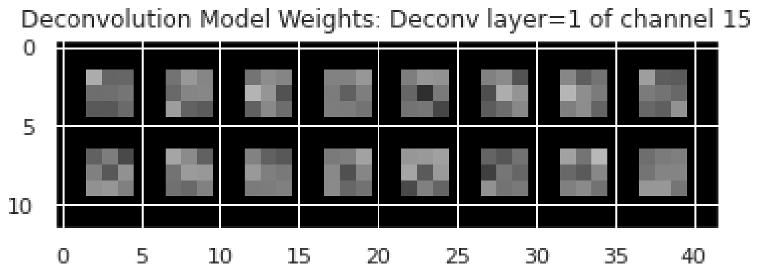


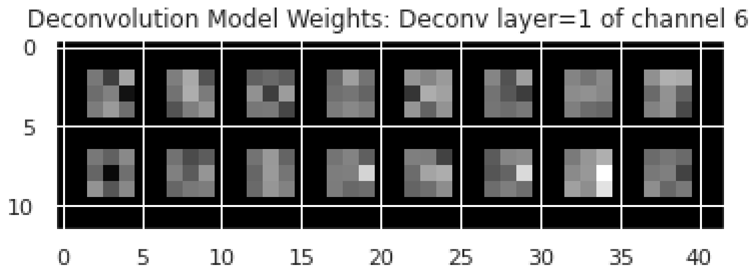
Figure 26: Decoder plot

5.3 Filters for Deconvolution Model

Conv1



((a))



((b))

Figure 27: Decoder plots

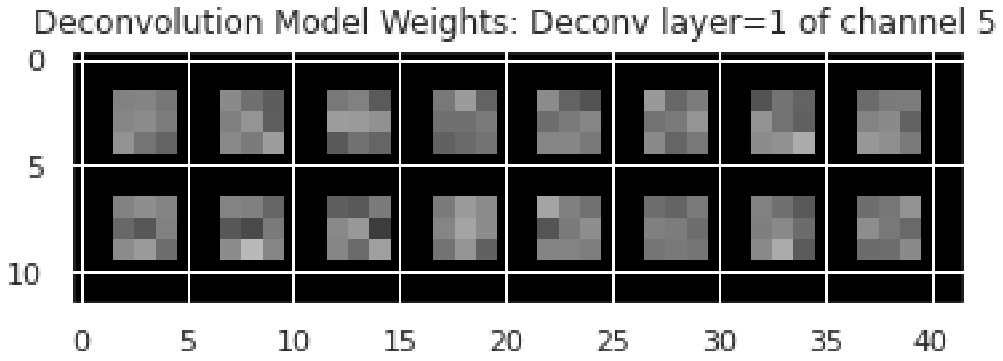
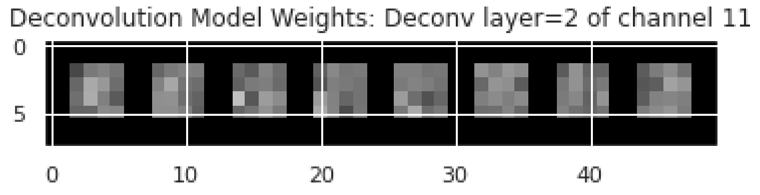
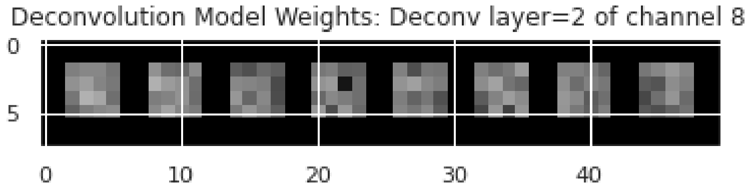


Figure 28: Decoder plot

Conv2



((a))



((b))

Figure 29: Decoder plots

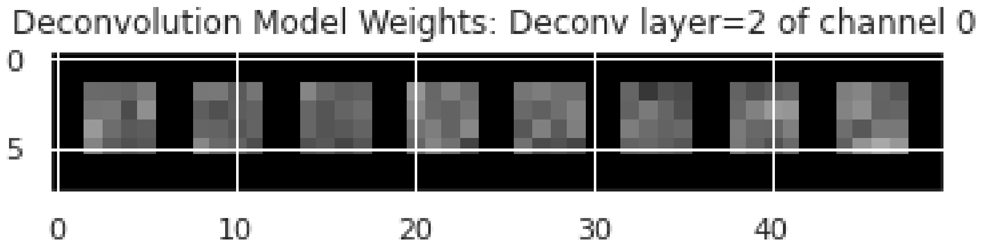
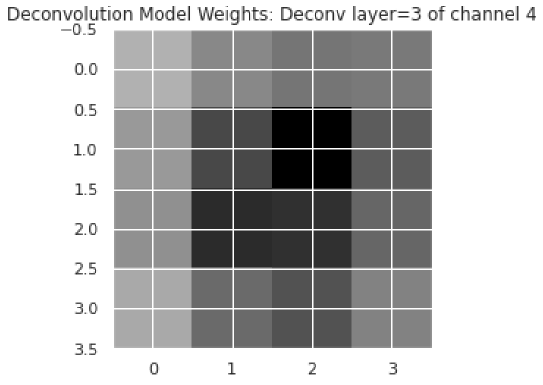
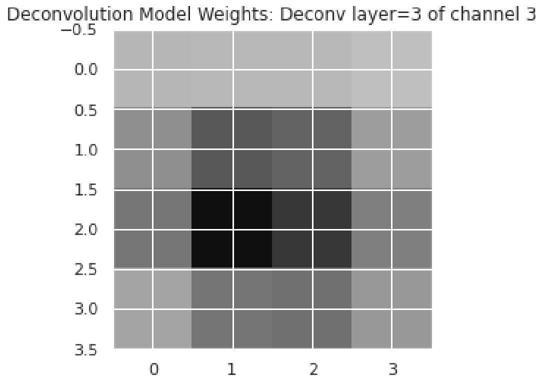


Figure 30: Decoder plot

Conv3



((a))



((b))

Figure 31: Decoder plots

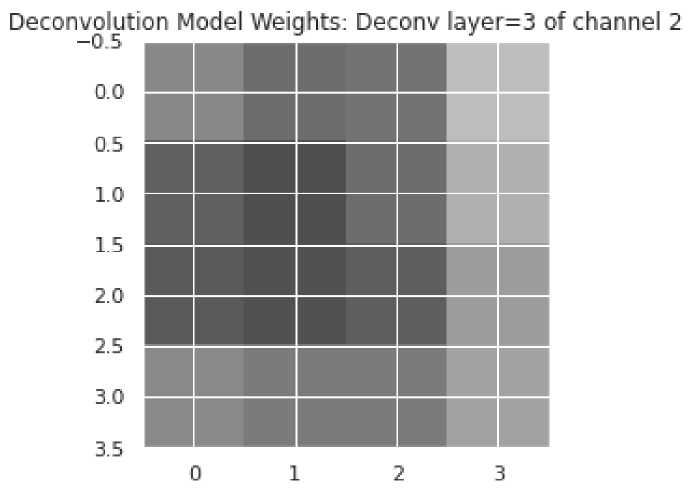
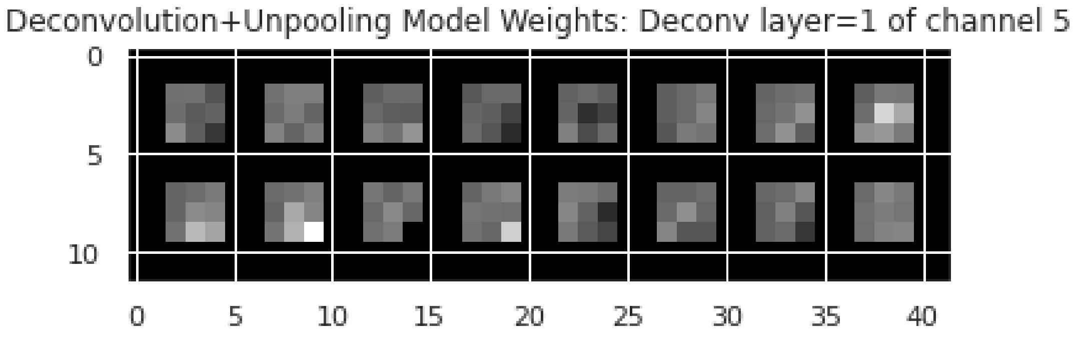


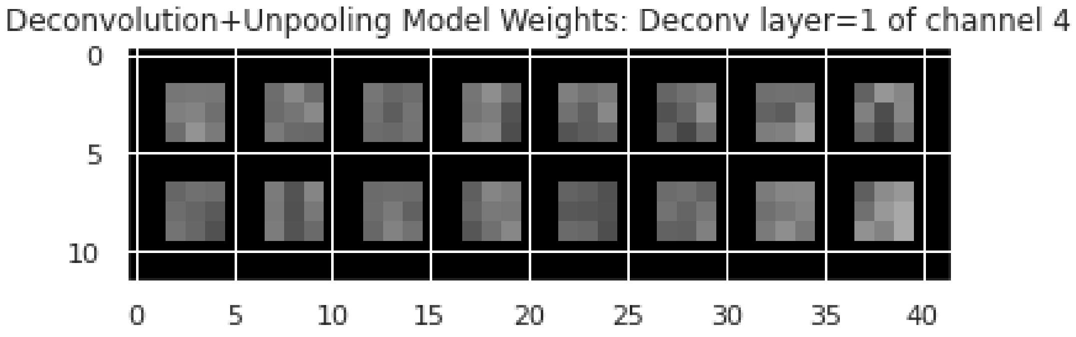
Figure 32: Decoder plot

5.4 Filters for Unpooling + Deconvolutional Model

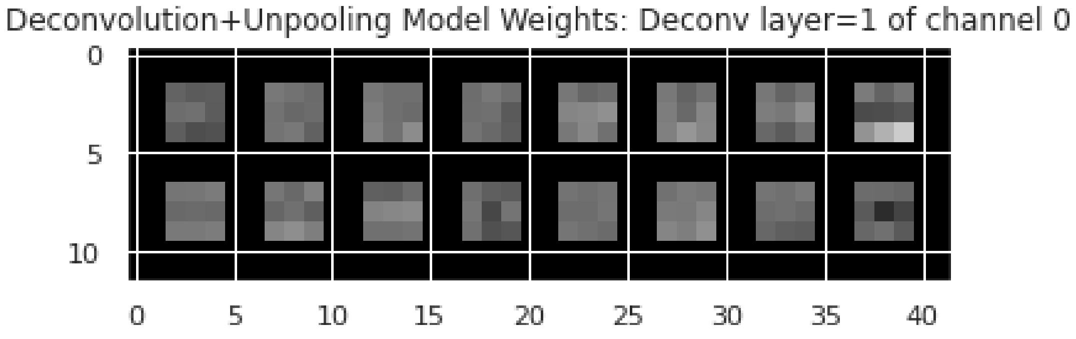
Conv1



((a))



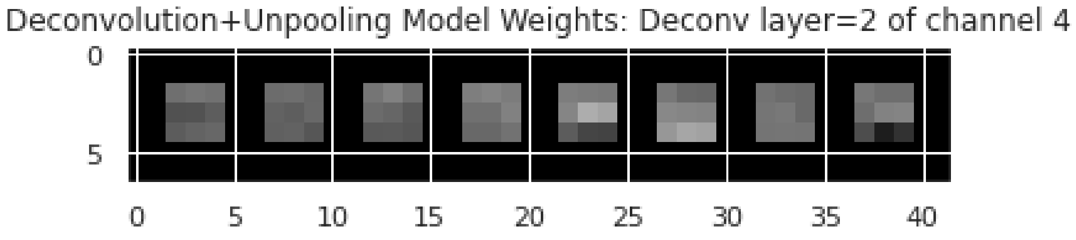
((b))



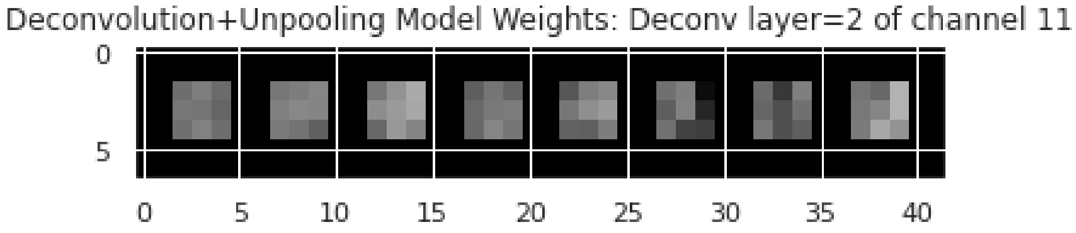
((c))

Figure 33: Decoder plots

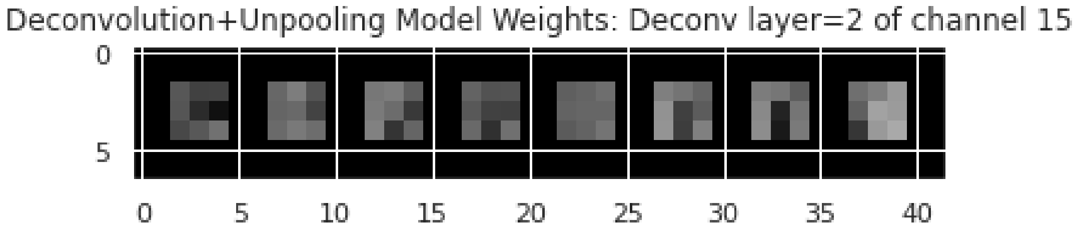
Conv2



((a))



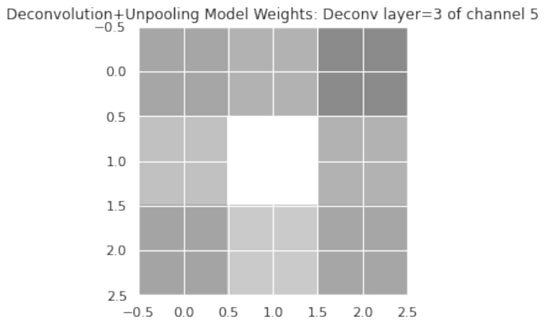
((b))



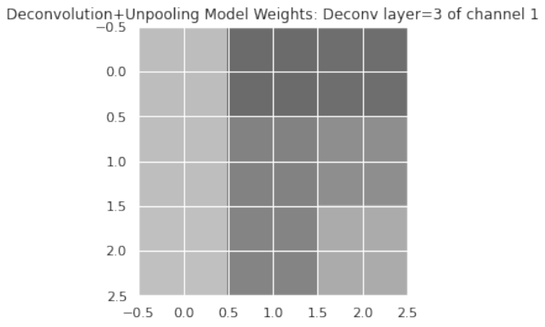
((c))

Figure 34: Decoder plots

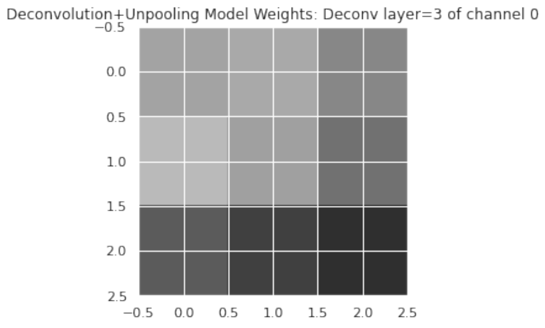
Conv3



((a))



((b))



((c))

Figure 35: Decoder plots

Inferences drawn from Convolutional Autoencoder

- From figure 24 it is evident that Deconvolution is better than Unpooling + Deconvolutional and Unpooling performs worse than all.
- The Deconvolutional autoencoder performs better than the Standard autoencoder as the convolutional layers learn local features that helps in learning and reconstructing.
- From Unpooling model, the channels in the beginning looks to contain meaningful image patters but as we go deeper in the channels the semantic information is lost as evident from the filters shown and model might not be able to learn subtle features of the image.
- Unlike Unpooling Model the Deconvolutional model learns features better as the deeper channels also seem to have patterns rather than just even distribution of pixels, which indicates that even deeper channels learn features.
- The Unpooling + Deconvoluional model performs midway between the other two. Deeper channels still seems to have meaningful features but some filters still gets even intensity all over and hence aren't able to strongly learn features which is evident from Figure 24 as the reconstructed output looks a bit scattered and seems to have more checker-board effect.