

Programming Assignment 4

R KAUSHIK — EE15B105

November 6, 2018

1 PCA and Auto Encoder comparison

Linear Auto Encoder

Mean Squared Error(MSE) of reconstructed images is used to compare the performance of PCA and Auto Encoder(AE) transformations. The Auto-Encoders were trained for untill convergence.

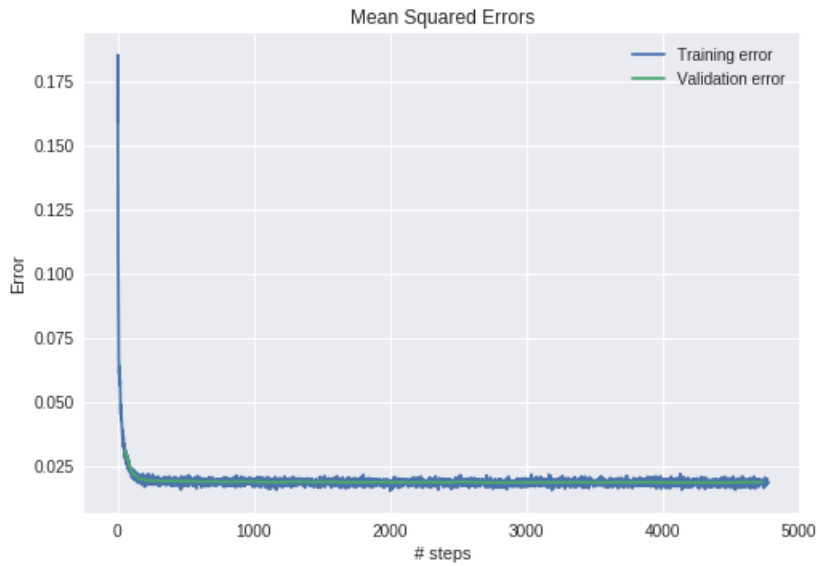


Figure 1: Plot showing training and validation error of linear Auto Encoder as training progresses

1. MSE of reconstructed images using AE with linear activation = **0.0182**
2. MSE of reconstructed images using PCA = **0.0177**

AE with linear activation would only impose linear relation between any two layers.

$$\mathbf{a}_i = \mathbf{W}\mathbf{a}_j$$

. Where \mathbf{W} is the weight matrix of dimension $len(a_j) \times len(a_i)$. Hence the relation between the input image and latent vector and that between latent vector and reconstructed images is linear. This is the case in Eigen Vector Decomposition (PCA) too. Hence AE with linear activation is similar in representational power to PCA. However, the linear AE cannot outperform PCA. It is because

- Transformation in PCA is precise and doesn't change if the same training data is fit again. The transformation done is optimal as it corresponds to the largest of eigen values.
- Decomposition in linear AE changes when the data is fit again. Gradient descent could not find the point of minima precisely. Hence there is a fall of performance when compared to PCA and the obtained latent representation need not be same at that obtained from PCA.

Standard Auto Encoder

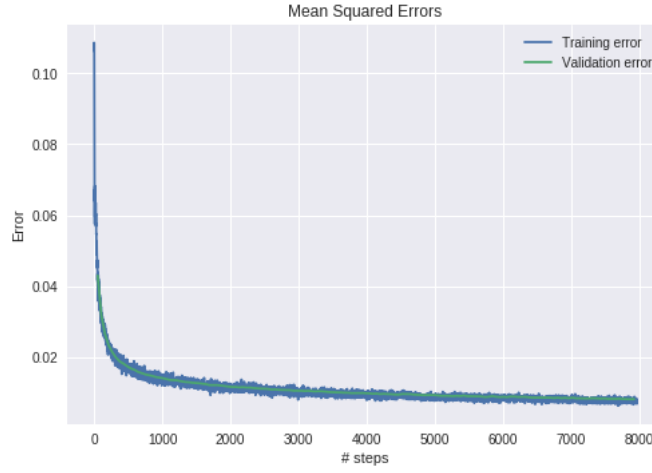


Figure 2: Convergence of training and validation error as training progresses for standard AE

MSE of reconstructed images using AE with linear activation = **0.0108** Non-linearity enable the AE to have the capacity to capture better or more representational details of the image. Hence its performance is better than PCA compression.

2 Standard Autoencoder

Hidden layer size	10	30	50	100	200	350
Reconstruction cost	0.0341	0.0179	0.0117	0.0058	0.0024	0.0007

Table 1: Reconstruction cost for different hidden layer sizes

2.1 Reconstruction of digit images

Trueimage	10	30	50	100	200	350

Table 2: Table shows the reconstructed images generated by AE having different hidden layer sizes (top of each image). The digit image from mnist test set were reconstructed using non-linear AE models trained in previous section

2.2 Reconstruction of other images

Trueimage	10	30	50	100	200	350

Table 3: Table shows the reconstructed images generated by AE having different hidden layer sizes (top of each image). The character image from emnist dataset were reconstructed using non-linear AE models trained in previous section

White Noise image reconstruction

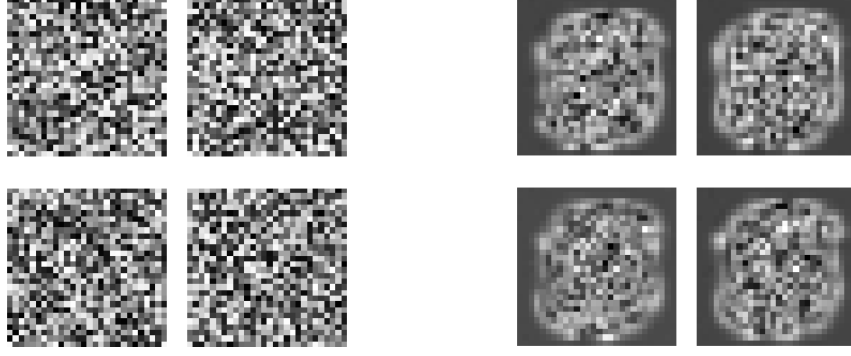


Figure 3: Reconstruction of noise images using AE with hidden layer size 350 - the best performing reconstruction model among all other models. The left image is the true image, right image is generated image from latent vector representation of true images

2.3 Visualizing latent vector

Hidden layer size	filter images									
10										
30										
100										
350										

Table 4: The table shows the visualization of filter values of hidden layer nodes. The first row has only 10 filters as the size of hidden unit is 10. For all other layers, the number of hidden layer nodes is at most 30. Hence the first 30 filters are visualized for all other models.

The filter images corresponding to models with lesser number of hidden layers (10 and 30) are more interpretable than filters of other models. As number of hidden layers increases, the filters become more and more complex.

From the filters that are interpretable, we could identify filters highlighting some specific numbers - such as filter in 4th column and 1st row of sets of filters corresponding to hidden layer size 10, represents number 7. We could almost map all the filters of 10-units hidden layer model to one of the 10 digits vaguely. This shows how the network has learnt to identify and segregate different digits from the input dataset.

Similar mapping could be identified in 30-unit Auto Encoder to some extent. But identifying such simple filters are quite impossible in AE's with more hidden units.

3 Over-complete AE with Sparsity Regularization

- **Performance of model**

The following table shows the performance of model for varying rates of weight of L1 loss

L1 loss weight	0.001	0.01	0.1	1.0	10.0	100.0
MSE of reconstructed image	5.1×10^{-4}	2.7×10^{-4}	4.54×10^{-4}	2.14×10^{-3}	0.0151	0.0597
Average value of latent vector	0.1131	0.0560	0.0199	0.0078	0.0044	0.0032

Table 5: The table shows the variation of MSE error and average of absolute value of latent vector activations as the L1 weight changes. We can see that as weight increases, the MSE of reconstructed image increases, indicating poorer performances. At the same time, the average value of latent vector activations decreases

The standard AE models had the average latent vector activations in order of 0.3

- **Filter visualization**

The Table 6 shows filters leading to latent vectors for models trained with different L1 loss weight. We observe that the filters corresponding to models trained with higher L1 weight are more sparse/blank/-plain and completely not interpretable.

4 Denoising Auto-Encoder

4.1 Behaviour of Standard AE with noisy image

Depending on the strength of noise and the size of latent vector layer, when the noise corrupted images were passed to the previously trained standard Auto-Encoder, the output image obtained was also a noisy version of the input image.

- The Fig 4 shows the input image and the reconstructed image for noise level of 0.1 ($\mathcal{N}(0, 0.1)$) reconstructed with a model with 350 unit hidden layer.
- An interesting observation was that only the portion of image containing the digit was reconstructed with noise. The surrounding areas of the reconstructed images were devoid of any noise.

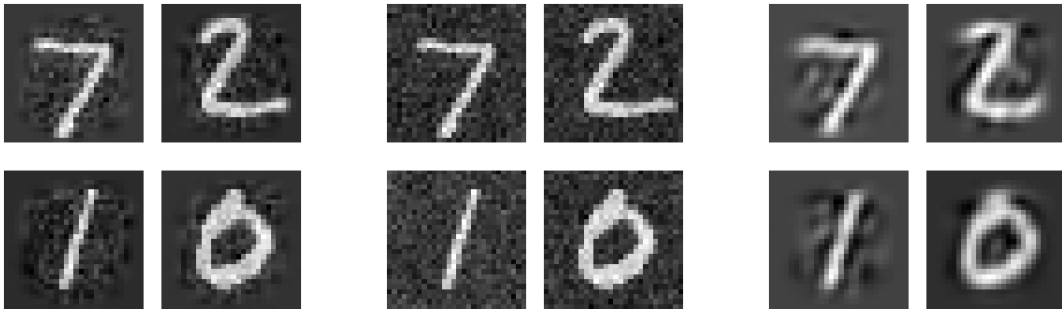


Figure 4: The figure shows the true image (corrupted with Gaussian White Noise) on the left. The other two images are reconstructed from model with 350 units and 50 units hidden layer respectively (from left to right). We can see that the reconstructed image contains noise only on the area where digit is present. the surrounding areas were cleaned of noise





















































































































































































L1 loss weight	filter images									
0.001										
										
										
0.01										
										
										
0.1										
										
										
1.0										
										
										
10.0										
										
										
100.0										
										
										

Table 6: The table shows the visualization of filter values of hidden layer nodes. The first row has only 10 filters as the size of hidden unit is 10. For all other layers, the number of hidden layer nodes is at most 30. Hence the first 30 filters are visualized for all other models.

- Judging by the order of reconstruction MSE (0.00792), we can say that the pattern of input noise is also reconstructed where ever the noise is present in the output.

4.2 Behaviour of standard AE with different noise levels

The following plot shows the variation of MSE with noise level in the input image.

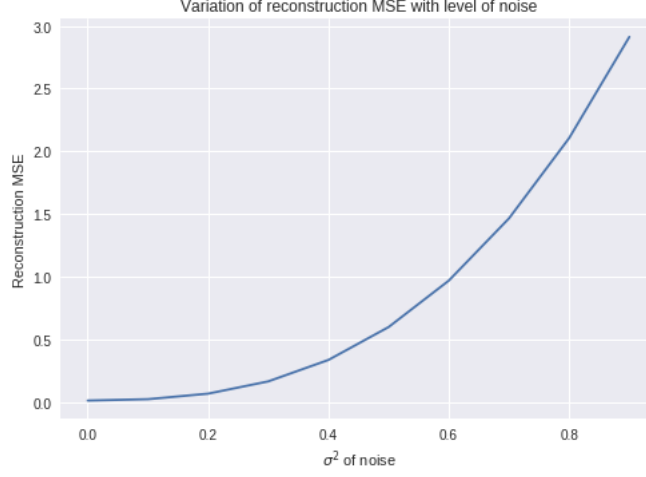


Figure 5: The plot shows the variation of MSE with noise level in the input image. The level of noise is measured in terms of variance of Gaussian distribution from which noise was generated and added to the input image. There is a quadratic increase in MSE and thereby drop in performance with noise level

For all levels of noise, one observation remains invariant. **The region surrounding the digits in the reconstructed images are devoid of noise from the input.** No matter how noisy the true image is, the region in which digit is not present would always be clean. The noisy patches would be confined to only regions on the digit. The following figure illustrates this.

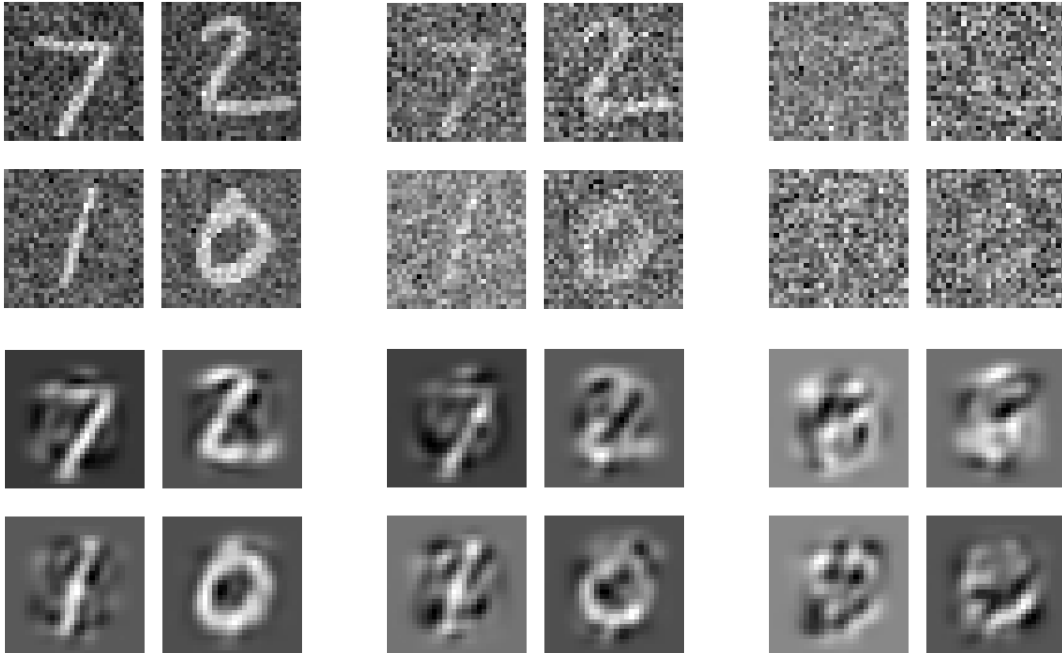


Figure 6: The figure shows the true image (corrupted with Gaussian White Noise) on the top and reconstructed image at the bottom. The left most image corresponds to $\sigma^2 = 0.2$. The middle image corresponds to $\sigma^2 = 0.5$. the right image corresponds to $\sigma^2 = 0.9$. The images were generated using the standard AE model with 100 hidden units

Even though the digit in the true image at noise level of 0.9 was completely obscured, However, the reconstruction is very poor.

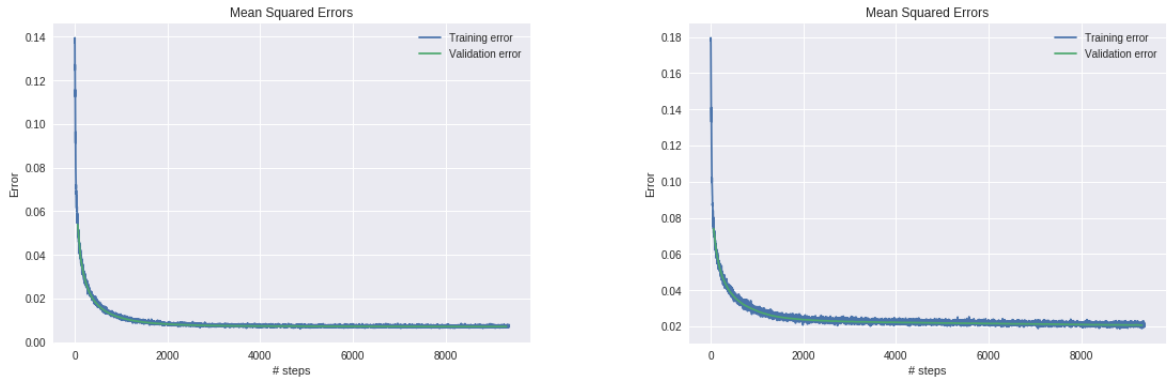


Figure 7: Training and test error convergence as training progresses. The plot on right corresponds to model trained with noise level 0.1. The right plot corresponds to model trained with noise level 0.5

4.3 Training Denoising Auto-Encoder

The figure 7 shows the training error convergence plot for two different noise level. The convergence plots for other noise levels looked similar.

Filter weights visualization and comparison

For ease of visualization, interpretation and comparison, feature maps of model with 10 hidden neuron was obtained (??).

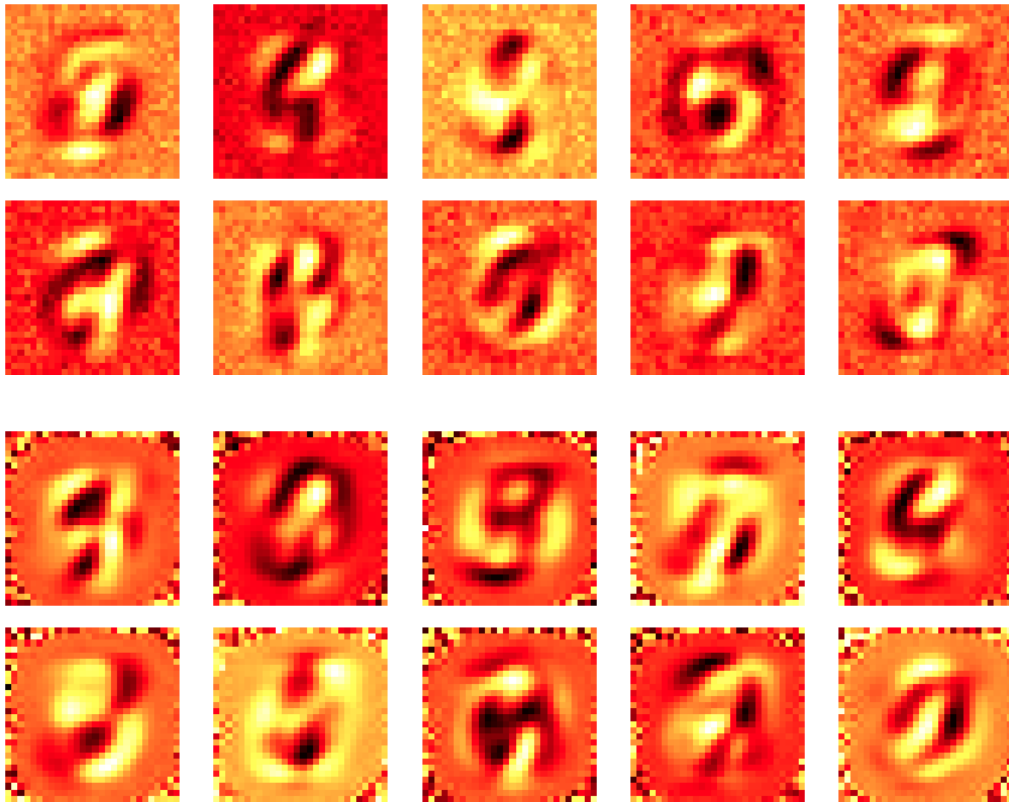


Figure 8: The figure shows the filters of Denoising AE (top) and Standard AE (bottom). Both the models had 10 hidden units

- The filters of DenoisingAE are more noisy when compared to StandardAE.
- The filters are less interpretable than StandardAE filters. For example the filter in 1st row and 1st column of top sub-figure and that in 1st row and 4th column of bottom sub-figure are similar. While the filter of the bottom image resembles digit 7 more, the filter in the top only vaguely gives the impression of 7.

5 PCA and DCT

1. **When Under-complete AE acts like DCT:** When the dataset contains images that have no semantic information. In such cases, it will be futile to try to extract semantic sense (like a person, a face, a tree, a mountain, etc that are usually present on natural images or a well defined pattern that can be extracted from the images of dataset) out of the image classes. Hence, data-independent compression would be a better way to compress images. For example, consider the case of compressing a signal. This signal could be anything like voice message, signals from sensors, seismic activity signals, etc. Messages such as those cannot be semantically segmented to meaningful components. So in such cases, the only representation which the AE could learn is the representation which could be obtained for any samples that are drawn *not* from any particular distribution. The latent representation would be independent of any distribution from which dataset is drawn. So in those cases, using AE to get latent representation could be same as applying DCT on the signals.

The datasets containing natural images or digit images are samples drawn from a particular distribution. In those cases AE would learn a low-dimensional manifold in which the data samples are concentrated. In those cases AE and DCT compression would be very different.

2. **over-complete AE and Dictionary learning:** Building a dictionary and representation corresponds to the optimization problem

$$\underset{\mathbf{D} \in \mathcal{C}, r_i \in \mathbb{R}^n}{\operatorname{argmin}} \sum_{i=1}^K \|x_i - \mathbf{D}r_i\|_2^2 + \lambda \|r_i\|_0 \quad \text{where } \mathcal{C} \equiv \{\mathbb{D} \in \mathbb{R}^{d \times n} : \|d_i\|_2 \leq 1 \ \forall i = 1, \dots, n\}$$

where $\mathbf{D} = [d_1, d_2, \dots, d_n]$ is the Dictionary and $\mathcal{R} = [r_1, r_2, \dots, r_K]$ is the representation.

While Sparse AE learns to optimize

$$\underset{\mathbf{W}_d \in \mathbb{R}^{d \times n}, r_i \in \mathbb{R}^n}{\operatorname{argmin}} \sum_{i=1}^K \|x_i - f(\mathbf{W}_d r_i)\|_2^2 + \lambda \|r_i\|_1 \quad \text{where } \mathcal{R} = f(\mathbf{W}_e X)$$

over the variables \mathbf{W}_d - weights of decoder and \mathbf{W}_e weights of encoder. Here X is the input data and $f(\cdot)$ is a non-linear function.

The differences are in the use of L^1 - norm as regularization term in Sparse AE instead on L^0 - norm, and non-linearity and use of non-linearity to generate latent representation and output.

6 Manifold learning

1. The subspace (manifold) in which the digit images are concentrated spans lesser dimensions than the input image itself. So when the noises are added to input images, the resulting image has **very less probability of remaining in the low dimensional manifold**. Hence when more noises are added to the input images, they would no longer represent digits. For example refer Figure 6.
2. On the other hand, when more noises are added to the latent representation of the image, the reconstructed image does not get invalidated easily. It is because the AE has learnt the decoder parameters which could transform the changes in latent representation to changes on the manifold without moving out of the manifold.

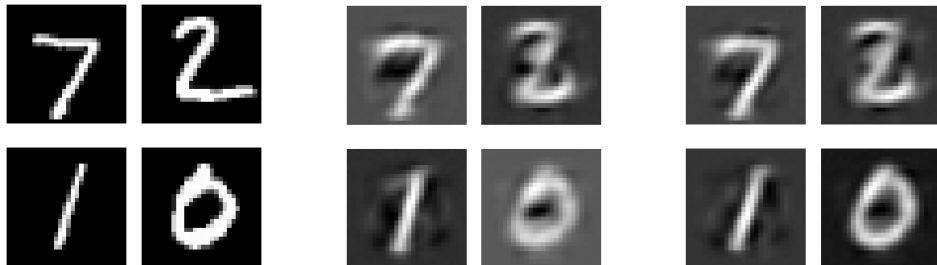


Figure 9: The left most image is the true image sampled from test set. The middle image shows the reconstruction obtained from latent representation. The right most image is reconstructed from latent representation of the true image after adding a small Gaussian White Noise. The reconstruction from noisy representation is not very different from the reconstruction obtained from un disturbed latent representation. This shows the extent to which the AE model has learnt the manifold representation

The reconstruction MSE of this model was found to be 2.573×10^{-2} .

7 Convolutional AE

7.1 Different types of upsampling

A Convolutional Auto-Encoder (CAE) was built with 3 convolutional layers, alternated by two pooling layers, followed by a fully connected layer in the encoder. The fully connected layers connected the features maps at the output of convolutional layers to 100 units to create the latent vector. The decoder architecture depended on the type of upsampling used, however, fully connected architecture remained the same for all the three types.

Note: While training the standard AE, the optimal value of learning rate was found to be 5×10^{-4} . Using learning rates above this value, didnt result in stable training. However, for CAE's the learning rate could be as high as 5×10^{-3} . So the best performing CAE's were better than Standard AE in terms of reconstruction MSE when optimal learning rates were used for training the models.

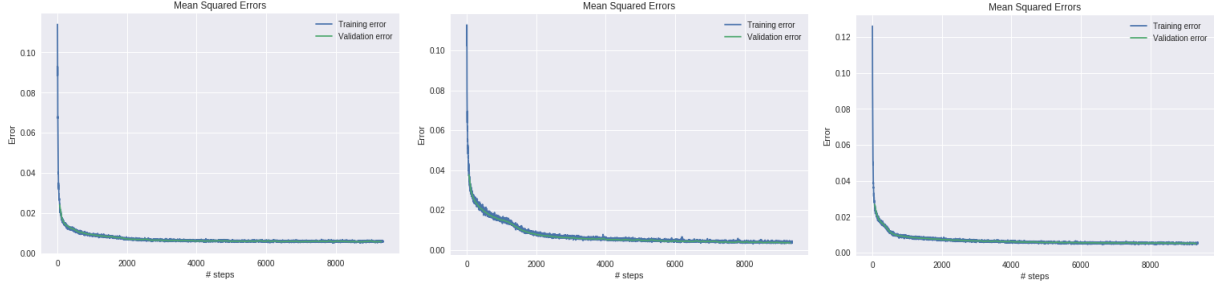


Figure 10: Training and validation error convergence plot for all 3 types of upsampling methods. The left-most plot corresponds to training of model with *unpooling* method of upsampling. The middle plot corresponds to *Deconvolution* method. The rightmost plot, to model with both *unpooling* and *Deconvolution* methods of upsampling

7.1.1 Unpooling

Layer #	Layer type	Input dimension	Output dimension	kernel size
1	unpool1	$4 \times 4 \times 16$	$7 \times 7 \times 16$	
2	conv1	$7 \times 7 \times 16$	$7 \times 7 \times 16$	3×3
4	unpool2	$7 \times 7 \times 16$	$14 \times 14 \times 16$	
3	conv2	$14 \times 14 \times 16$	$14 \times 14 \times 8$	3×3
4	unpool3	$14 \times 14 \times 8$	$28 \times 28 \times 8$	
5	conv3	$28 \times 28 \times 8$	$28 \times 28 \times 1$	3×3

Table 7: The architecture of decoder using only *unpooling* for upsampling the feature maps

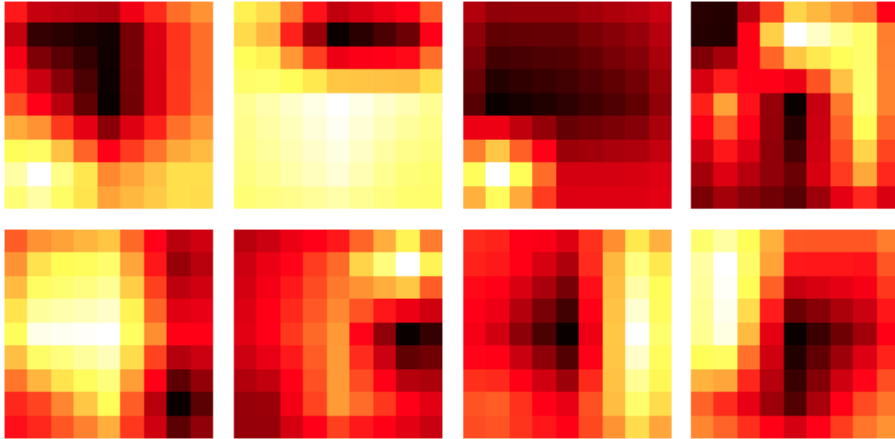


Figure 11: Filter weights of conv2 unpool layer. The real dimension of the filter image is 3×3 . However, for visualization purposes, the images are resized to 9×9

MSE of reconstructed images of test set was 5.722×10^{-3}

7.1.2 Deconvolution

Layer #	Layer type	Input dimension	Output dimension	kernel size
1	deconv1	$4 \times 4 \times 16$	$7 \times 7 \times 16$	3×3
2	deconv1	$7 \times 7 \times 16$	$14 \times 14 \times 8$	3×3
3	deconv2	$14 \times 14 \times 8$	$28 \times 28 \times 1$	3×3

Table 8: The architecture of decoder using only *Deconvolution* method for upsampling the feature maps

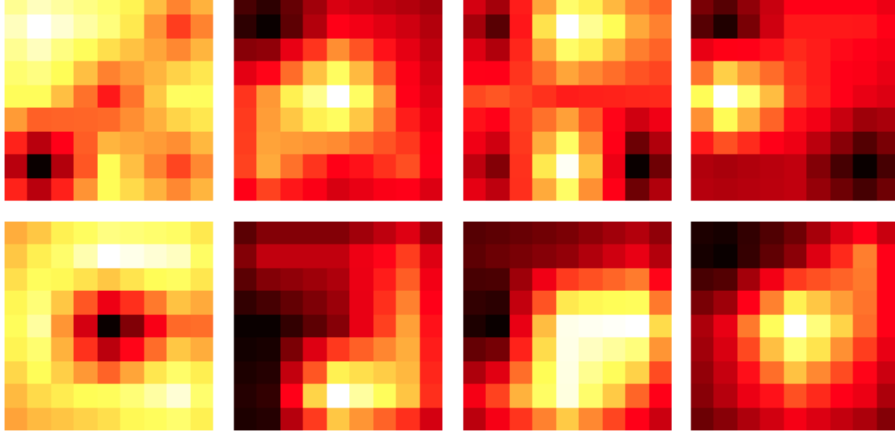


Figure 12: Filter weights of deconv2 unpool layer. The real dimension of the filter image is 3×3 . However, for visualization purposes, the images are resized to 9×9

MSE of reconstructed images of test set was 3.601×10^{-3}

7.1.3 Unpooling and Deconvolution

Layer #	Layer type	Input dimension	Output dimension	kernel size
1	unpool1	$4 \times 4 \times 16$	$7 \times 7 \times 16$	
2	conv1	$7 \times 7 \times 16$	$7 \times 7 \times 16$	3×3
4	unpool2	$7 \times 7 \times 16$	$14 \times 14 \times 16$	
3	conv2	$14 \times 14 \times 16$	$14 \times 14 \times 8$	3×3
5	deconv1	$14 \times 14 \times 8$	$28 \times 28 \times 1$	3×3

Table 9: The architecture of decoder using both *unpooling* and *deconvolution* for upsampling the feature maps

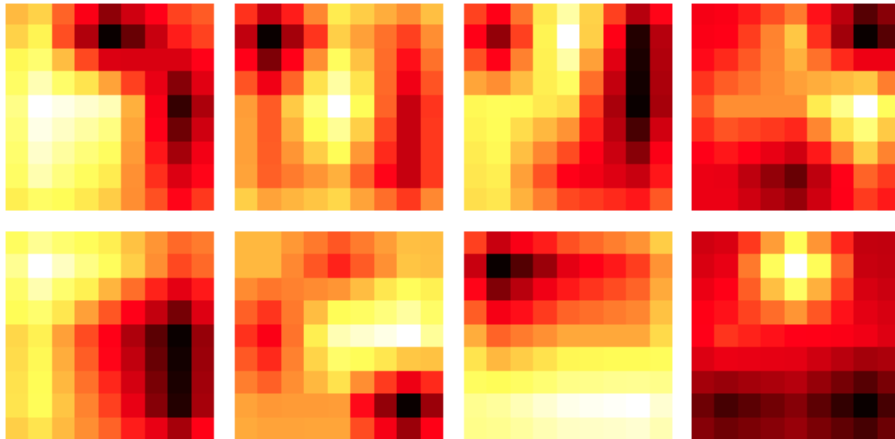


Figure 13: Filter weights of conv2 unpool layer. The real dimension of the filter image is 3×3 . However, for visualization purposes, the images are resized to 9×9

MSE of reconstructed images of test set was 5.267×10^{-3}

7.2 Reversing order of pooling and convolution

When the order of pooling and convolutional layers are reversed, the performance degrades. It is because there was a loss of rich information due the pooling layer, which was applied before convolution layers could extract useful low-level features fresh from the input. The pooling layers reduce the resolution of the feature maps for the next convolutional layers. Hence the loss of information had resulted in the loss of performance of the model in reconstructing the input image.

The reconstruction MSE on test images was obtained to be 2.606×10^{-2}

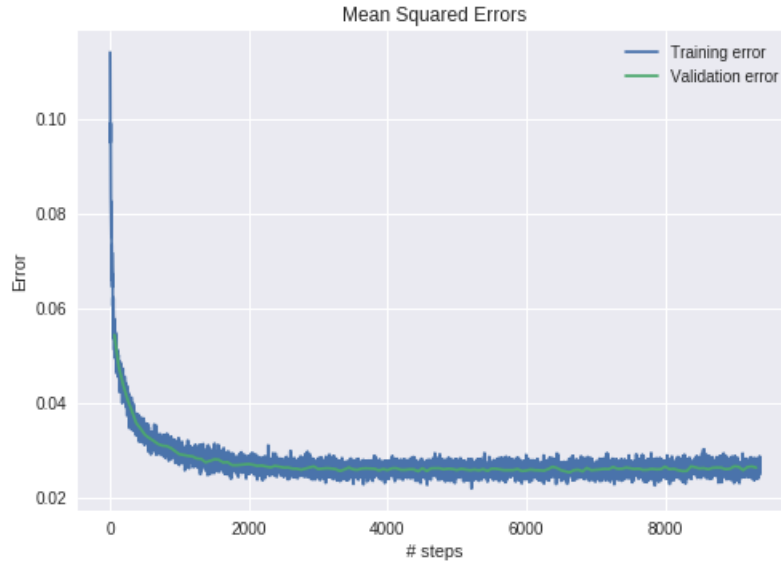


Figure 14: Training and validation error convergence plot for CAE with pooling layer occurring before convolutional layer