UPPSALA UNIVERSITET

COMPLEX DATA: ANALYSIS & VISUALISATION

Exercise 3
Deep Learning

EFTHYMIA CHANTZI

MSc in Bioinformatics

Email: Efthymia.Chantzi.0787@student.uu.se

January 4, 2016

MATLAB R2015b and the Neural Network Toolbox^{TM} [1] is used for the fulfillment of this assignment. The documented code that pertains to the implementation of the following tasks is accessible under Deep-Learning/Exercise3 at:
`https://github.com/EffieChantzi/Deep-Learning.git`.

In this assignment, *deep networks* are trained with the goal of image reconstruction. Two different approaches associated with the initialization of weights and biases are employed; *principle component analysis* and *autoencoders*. Both methods are implemented and tested against each other. The training and test datasets consist of 5000 digit images with 784 pixels (digittrain_dataset, digittest_dataset).

Provided that two different implementations of deep learning must be compared, common training settings are used, which are stated in table (1):

Training Settings	
Algorithms	
Data Division	Random (*dividerand*)
Training Function	Scaled Conjugate Gradient (*trainscg*)
Performance Function	Mean Square Error (*mse*)
Termination Criteria	
Maximum Number of Training Iterations (*max_epochs*)	1500
Minimum Gradient Magnitude (*min_grad*)	10^{-6}
Maximum Number of Validation Increases (*max_fail*)	6

Table 1: *Common training settings and parameters used for all deep networks of this exercise*

1 Task A

In this task, PCA[1] is applied to the training set (digittrain_dataset), in order to be compressed from 784 to M dimensions, which result in a pleasant visual inspection after reconstruction.

The two following figures illustrate the result of PCA compression on the five first images of the training dataset, using $M = 40$ and $M = 50$, respectively. Additionally, the *total* mean squared error (E_M) is displayed on the title. With the term *total*, the mean reconstruction error of *all* images in the training set is indicated, despite the fact that on the figure only five images are displayed.

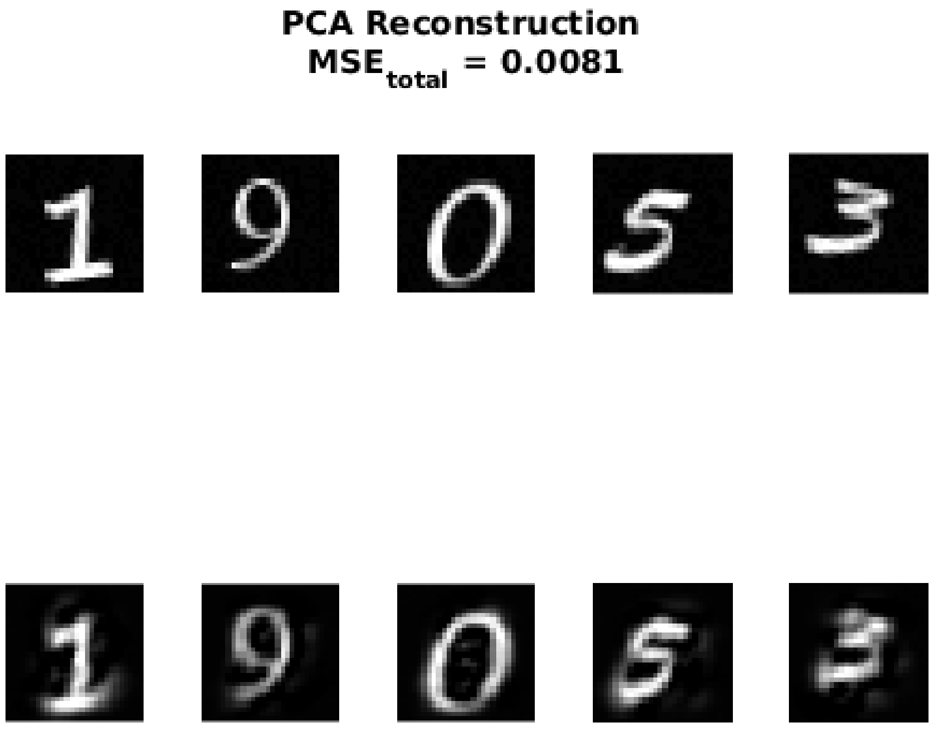


Figure 1: 40 *Principal Components* $\longmapsto E_M = 0.0081$

[1]Principal Component Analysis

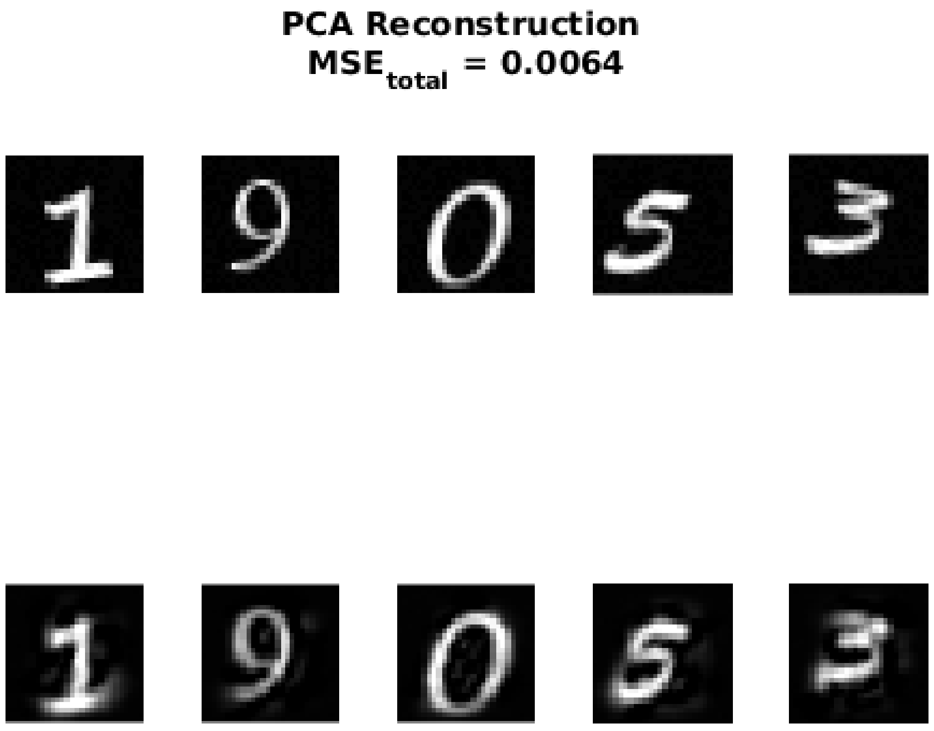


Figure 2: *50 Principle Components* $\longmapsto E_M = 0.0064$

It is true that both values of M are associated with visually acceptable and successful reconstructions of the original images, since they accompanied with remarkably small total mean reconstruction errors.

2 Task B

In this task, it is investigated whether a deep "bottle-neck" neural network with three hidden layers can be used to compress the images down to fewer dimensions than $M = 50$, as obtained with PCA in the previous question, while maintaining the same total mean squared error.

50-15-3-15-50

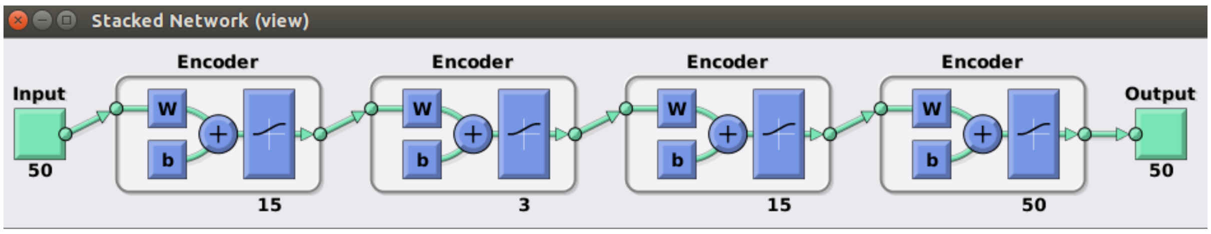


Figure 3: *Bottle-neck Neural Network 50-15-3-15-50*

Time	
Pre-training	00:03:03
Fine Tuning	00:01:16

Table 2: *Time requirements on local machine*

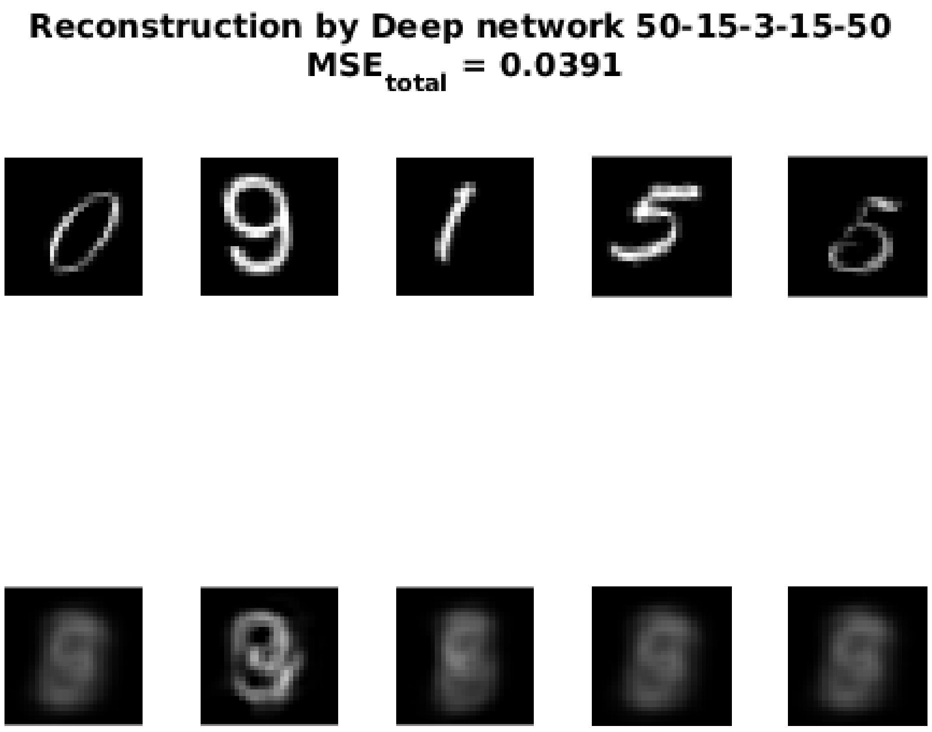
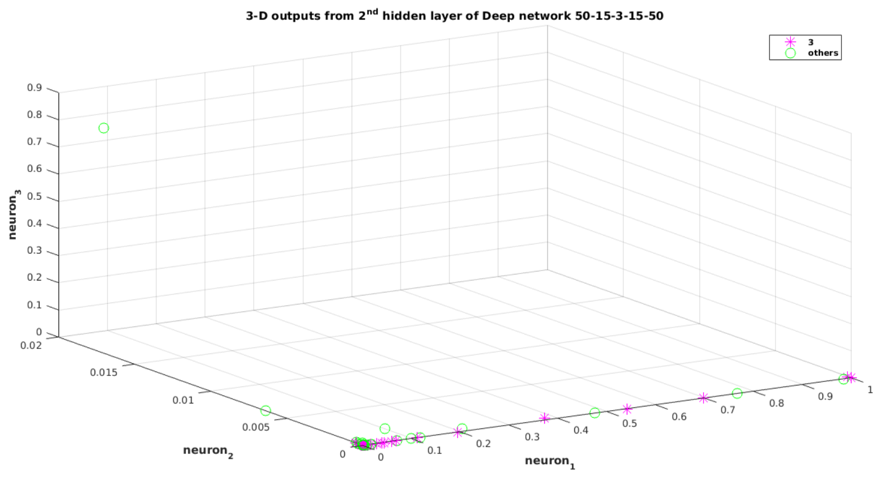


Figure 4: *Reconstruction by Bottle-neck Neural Network 50-15-3-15-50* $\longmapsto E_M = 0.0391$



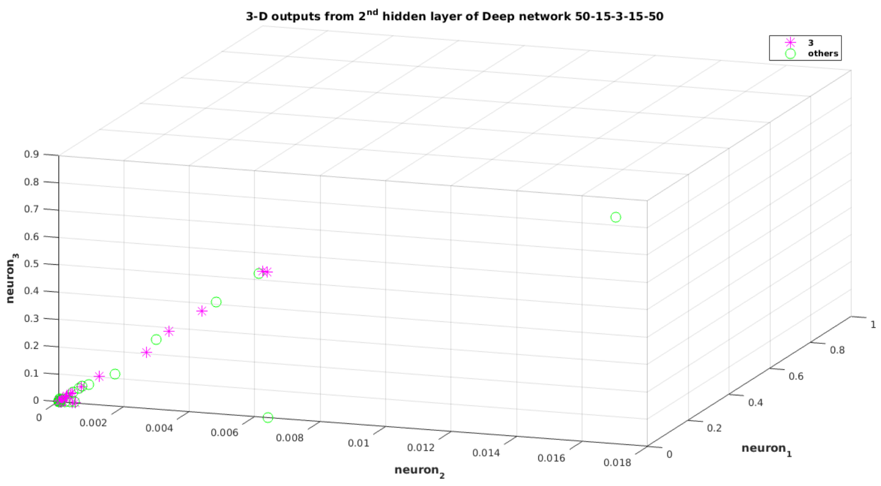


Figure 5: *2^{nd} layer response on 200 randomly selected images: 100 showing "3" and 100 showing "other"*

50-20-10-20-50

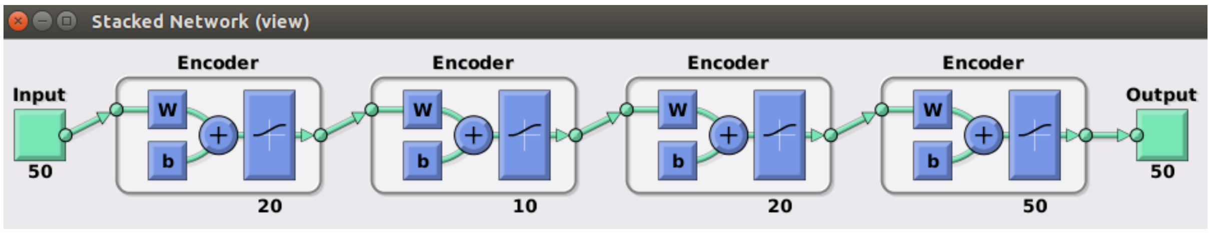


Figure 6: *Bottle-neck Neural Network 50-20-10-20-50*

Time	
Pre-training	00:04:21
Fine Tuning	00:01:59

Table 3: *Time requirements on local machine*

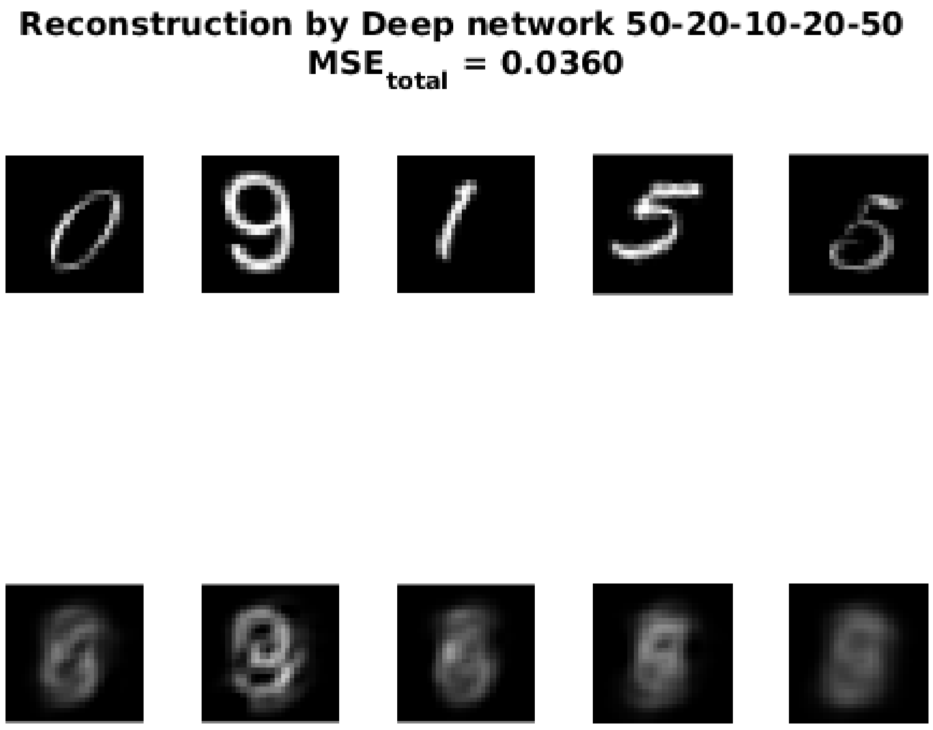


Figure 7: *Reconstruction by Bottle-neck Neural Network 50-20-10-20-50* $\longmapsto E_M = 0.0360$

Judging by the obtained results, the same level of reconstruction error cannot be retained by compressing further the images, using stacked autoencoders. Even if more neurons are introduced in the hidden layers, 10 in this case, although the total mean squared error decreases from 0.0391 to 0.0360, it is still significantly larger than that one of PCA; 0.0064.

Furthermore, in the description of the exercise, the use of linear nodes in the output layer is required. However, the function *trainAutoencoder*[2] allows only two different options regarding the transfer function for the encoder; *logsig* and *satlin*. The latter one produces worse results compared to the former one and thus, *logsig* is eventually used.

3 Task C

In this part, weights and biases of a deep neural network with structure 784-25-3-25-784, are initialized based on PCA. This initialization procedure is applied on two different categories of neurons' activation function; *logistic sigmoid* (logsig) and *hyperbolic tangent sigmoid* (tansig) function. The latter one is requested as a separate question in task G, but its implementation is nested in this task.

The results are presented in detail in the next section that pertains to task D. It is important though to mention that setting $s = \frac{1}{3}$ yields larger reconstruction errors compared to smaller values(i.e., $\frac{1}{20}$, $\frac{1}{40}$).

4 Tasks D, E, G

4.1 Deep network (784-25-3-25-784) with PCA initialization

4.1.1 logsig

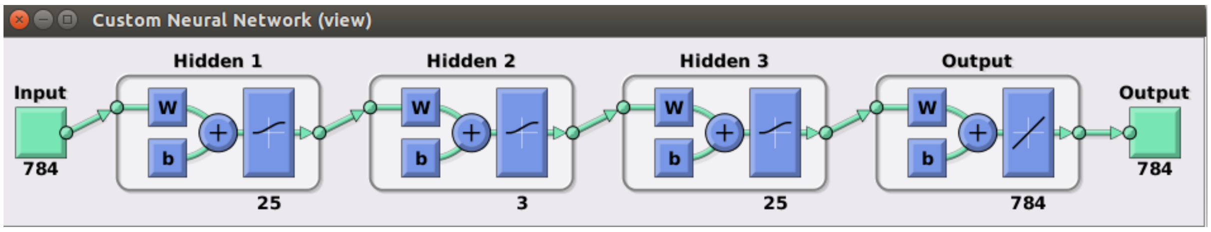


Figure 8: *Deep Neural Network 784-25-3-25-784 with PCA initialization*

Time	
Training	00:04:22

Table 4: *Time requirements on local machine*

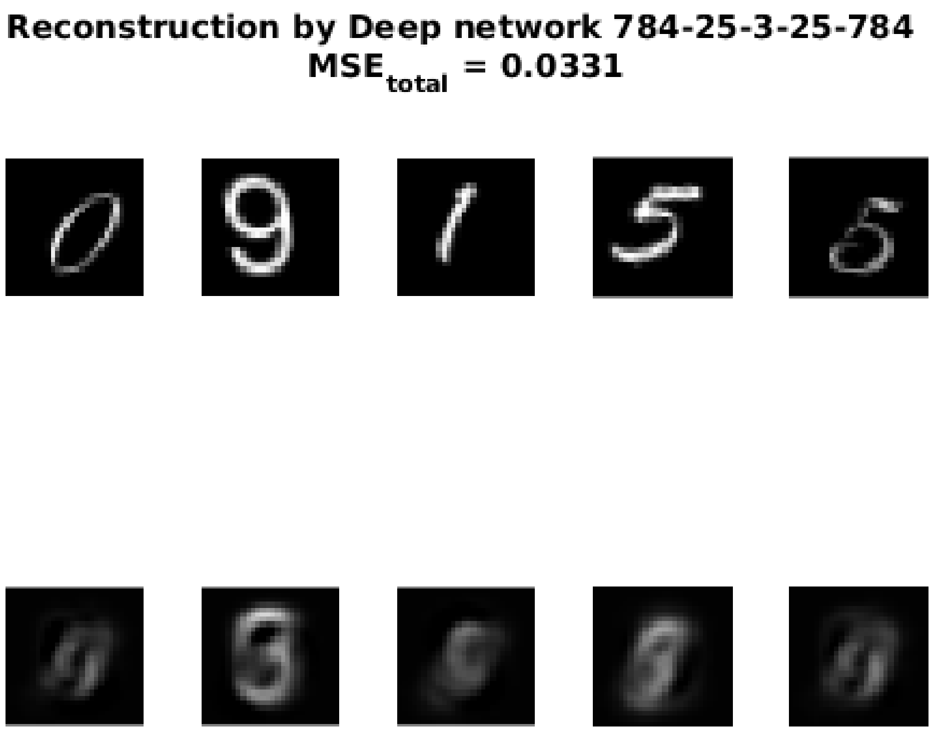
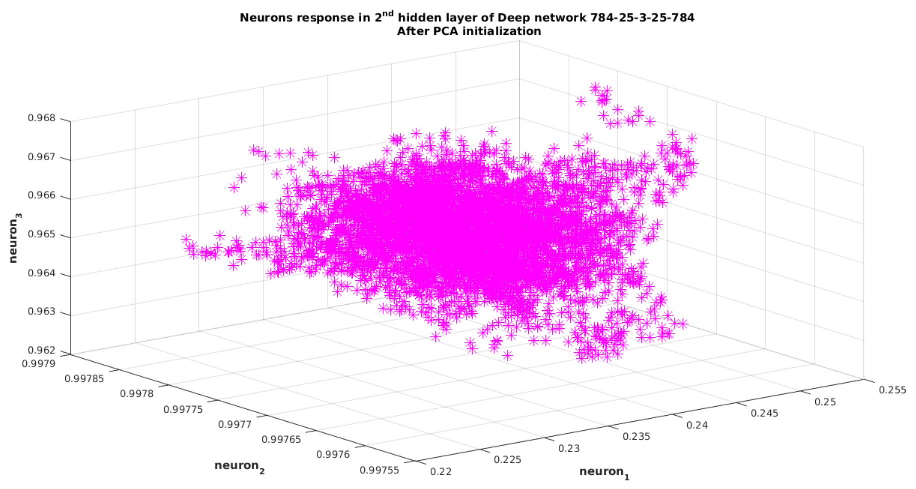


Figure 9: *Reconstruction by Neural Network 784-25-3-25-784 after PCA initialization* $\longmapsto E_M = 0.0331$



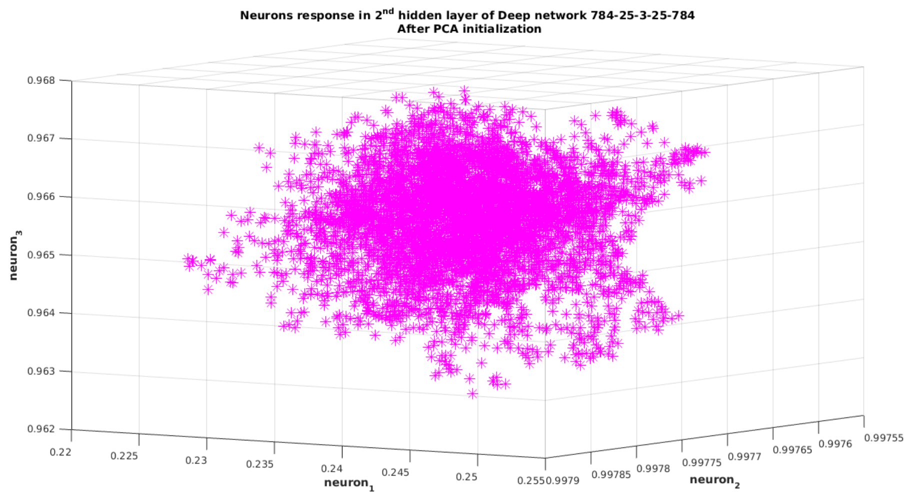


Figure 10: *2^{nd} layer response on the whole test dataset*

4.1.2 tansig

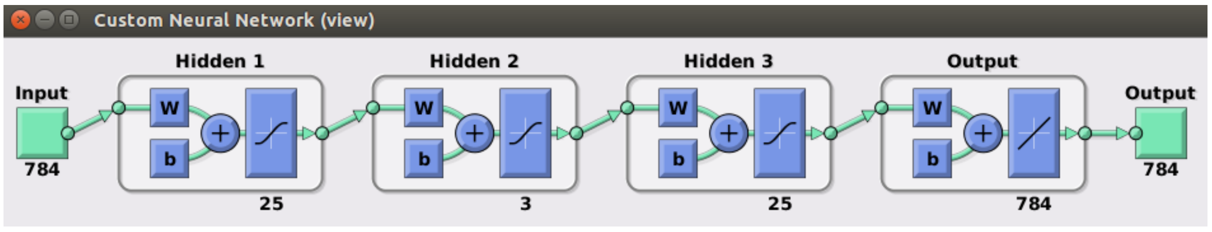


Figure 11: *Deep Neural Network 784-25-3-25-784 with PCA initialization*

Time	
Training	00:02:44

Table 5: *Time requirements on local machine*

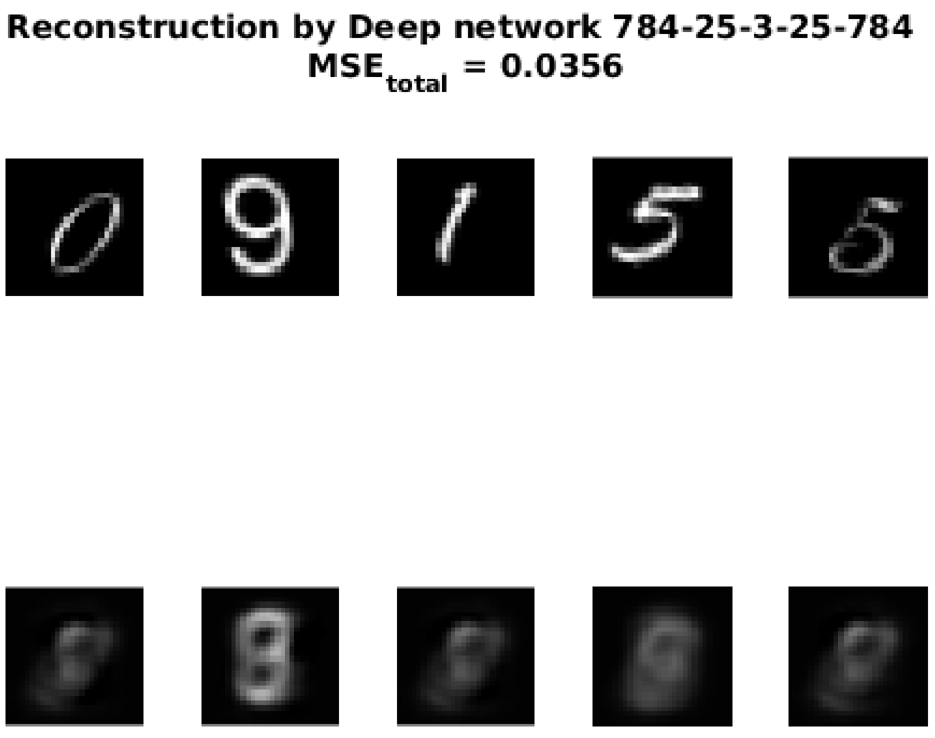
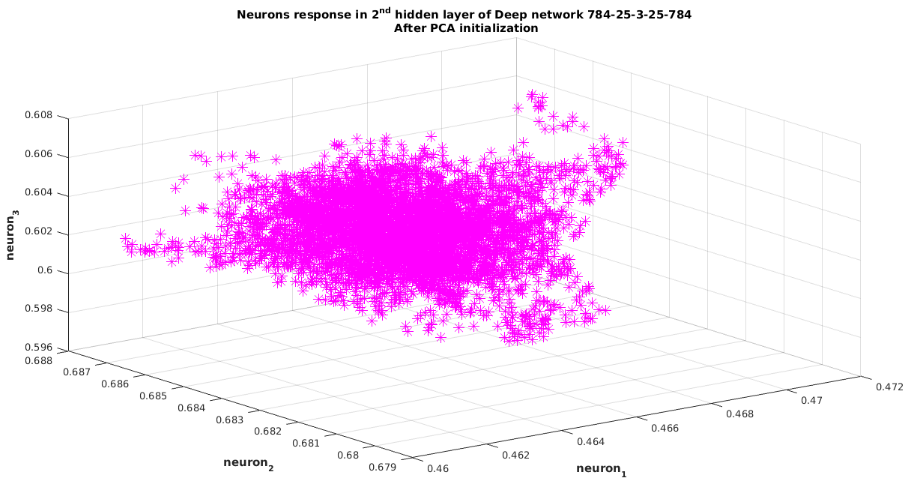


Figure 12: *Reconstruction by Neural Network 784-25-3-25-784 after PCA initialization* $\longmapsto E_M = 0.0356$



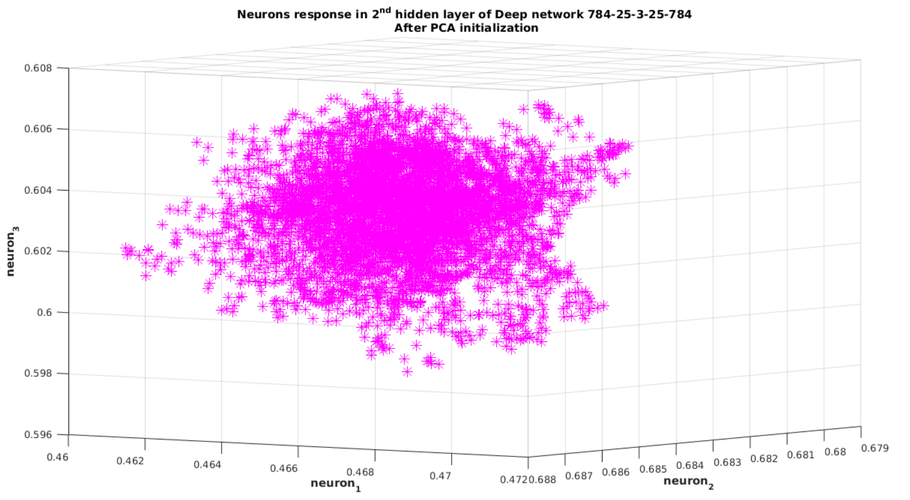


Figure 13: *2^{nd} layer response on the whole test dataset*

4.2 Deep network (784-25-3-25-784) with autoencoders initialization

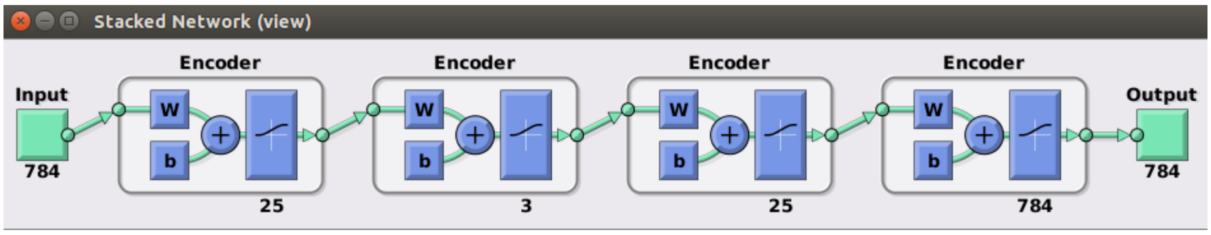


Figure 14: Deep Neural Network 784-25-3-25-784 with autoencoder initialization

Time	
Pre-training	00:26:45
Fine Tuning	00:19:30

Table 6: Time requirements on local machine

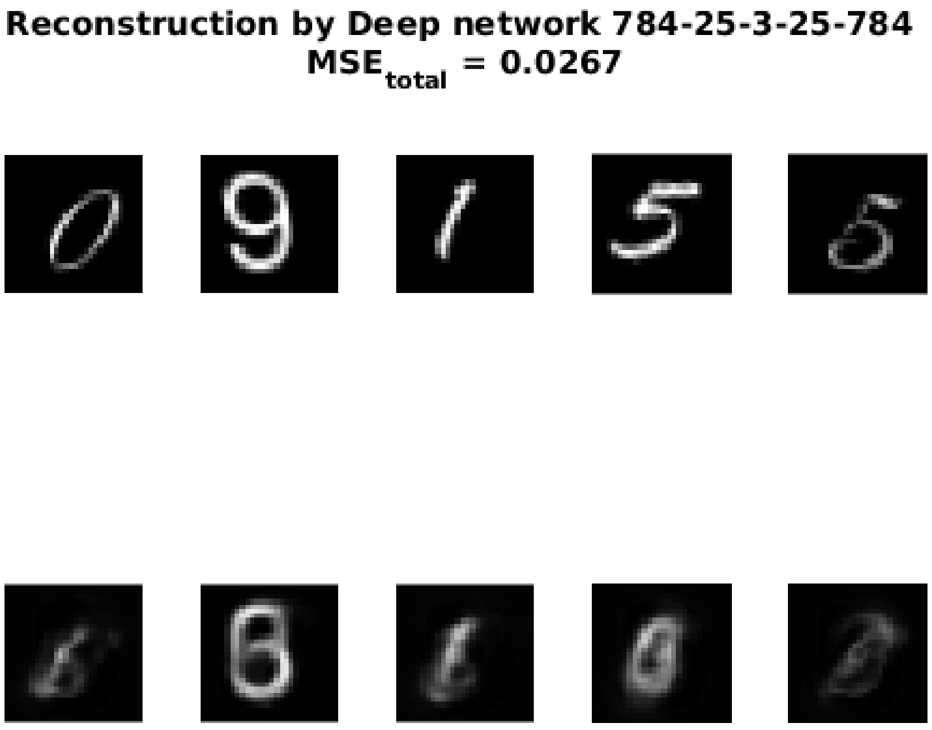


Figure 15: Reconstruction by Neural Network 784-25-3-25-784 after autoencoder initialization $\longmapsto E_M = 0.0267$

It is apparent that using PCA initialization speeds up remarkably the neural network learning procedure compared to the employment of stacked autoencoders, since the prolonged pre-training of the separate autoencoders is avoided. However, the total mean squared error using stacked autoencoder initialization is smaller compared to that one obtained by PCA based initialization.

5 Task F

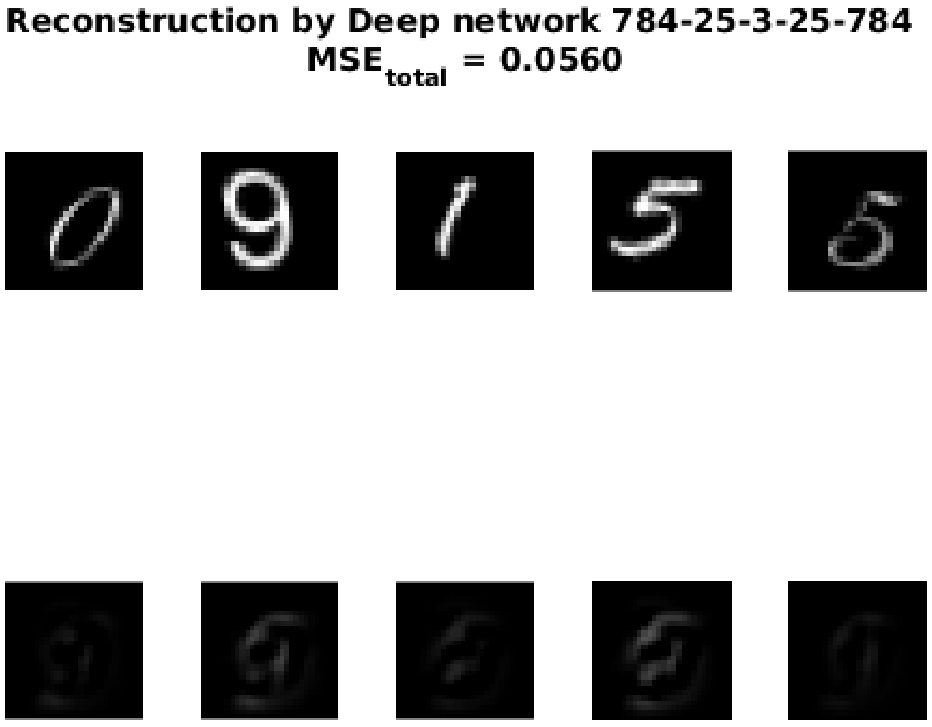
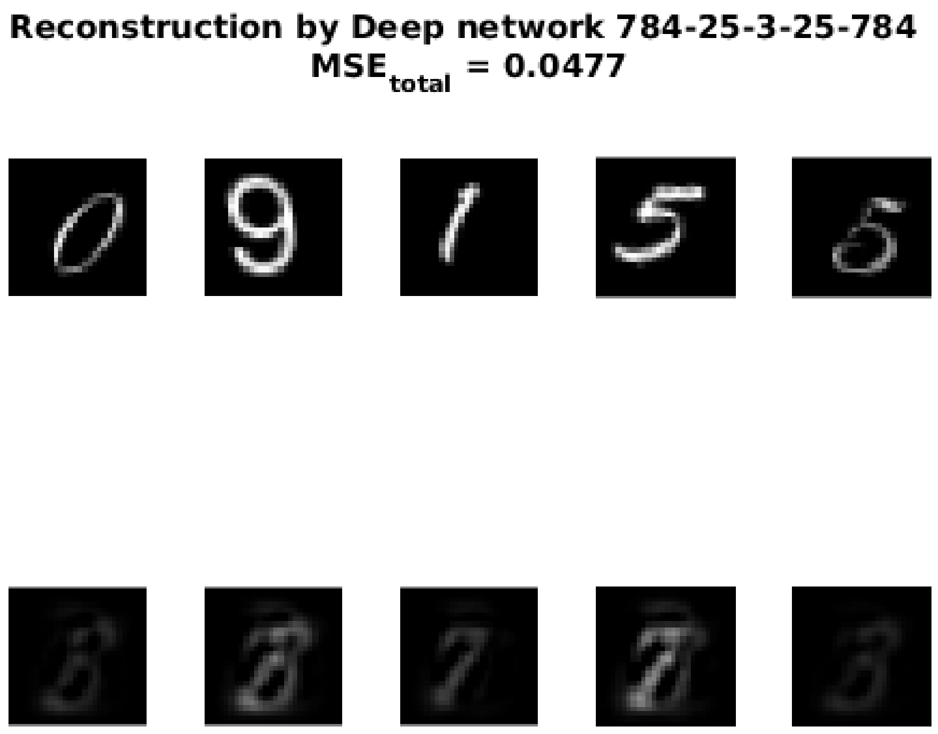


Figure 16: *Reconstruction by Neural Network 784-25-3-25-784 after PCA initialization and only* 200 *training images* $\longmapsto E_M = 0.05604$

It is obvious that having a limited number of training images restricts the learning procedure and thus, the network ends up memorizing the training examples rather than learning to generalize to new situations. In this case, the performance is poor with total mean square error $E_M = 0.05604$, due to the fact that only 200 training examples are used, from which one hundred is “3”. Consequently, when the trained network is tested on new unseen data from the test dataset, it performs poorly.

In Matlab, the built-in procedure of early stopping is activated by default, which means that when the validation error increases for a few consecutive times (max_fail = 6), then this is an indication of overfitting the data and hence, the training is terminated. In this case, the maximum number of consecutive validation increases is reached too early and the training is forced to terminate. As a result, overfitting is avoided but still the learning procedure is premature to provide a sufficient reconstruction of the test data and for this reason, a large value for the total mean reconstruction error is gained.



In this case, the built-in procedure of early stopping is deactivated by setting its value to a much greater value than the default (i.e., 100). In this way, overfitting is invoked in some sense, and this fact is verified both by the poor reconstruction performance ($E_M = 0.0477$) and visual inspection. It is apparent that the network has memorized the training examples, where there are a lot of “3” and “7” images, since on the test data the reconstruction produces the digit “3, 7” very clearly on completely different input images.

References

[1] “MathWorks Neural Network Toolbox.” http://se.mathworks.com/products/neural-network/. Accessed: November 20, 2015.

[2] “trainAutoencoder.” http://se.mathworks.com/help/nnet/ref/trainautoencoder.html#responsive_offcanvas. Accessed: November 21, 2015.