UPPSALA UNIVERSITET

COMPLEX DATA: ANALYSIS & VISUALISATION

Exercise 5

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 along with the datasets that pertain to the implementation of the following tasks are accessible under Deep-Learning/Exercise5 at:
https://github.com/EffieChantzi/Deep-Learning.git.

1 Task A

In this task, compression using *principal component analysis* and *deep neural networks* is applied to the second set of supplementary time-lapse microscopy movies provided by the article "*Phenotypic profiling of the human genome by time-lapse microscopy reveals cell division genes*" [2]. This second set of supplementary movies(16-30) contains only the highlighted single cell from each one of the movies in the first set(1-15).

The aforementioned time-lapse microscopy movie dataset requires pre-processing, so that it can be used as a d-dimensional matrix of x_n vectors, where each vector represents an extracted subimage and d is equal to the number of pixels(dimensions) of these subimages, which are in turn extracted from the frames of the provided movies.

As already mentioned, the second set of supplementary movies, which is used for this assignment, includes videos, whose frames are centered around the respective highlighted single cell from the first set of supplementary movies. This actually means that the frames from each video would most probably contain useful information and visual features in an area around the center. This centered region helps us minimize the risk of selecting subimages that do not display any useful information and hence, are useless for the learning procedure. Although this approach minimizes the selection of uninformative subimages that are caused by an arbitrary way of selection, there would still be a small number of them, since the frames represent consecutive scenes of cell division genes, which change positions and forms over time. This problem is tackled by setting a threshold for the minimum meaningful average intensity that an image should have in order to be considered rich in terms of its representation. All images, whose mean intensity is smaller than this threshold are extracted from the final dataset. In this way, the extracted subimages are filtered, but the number of learning examples is reduced indicating higher chances of overfitting and poor predictive performance on new unseen data.

Furthermore, two more important issues that should be taken into consideration are related to the number and dimensions of frames. Being more specific, each video consists of different number of frames, while the dimensions of frames among the videos are differentiated. Thus, the position of the extracted patch from all the selected frames of each movie must be dynamically adjusted to the respective dimensions in accordance with the patch size. As far as the number of frames is concerned, it is found that some videos have 93 frames, while others have 94 or 95. Under these circumstances, the extracted number of frames per movie is set to 93. This indicates that the total number of images in the final dataset would be $(\#\text{ movies} \times \#\text{ frames}) = 93 \times 15 = 1395$, before the filtering procedure. However, after the filtering, the total number of images is reduced to 1274.

In terms of the different frames' size and provided that the extracted subimage is a 40×40 matrix, there should be some kind of dynamic method that decides the position of the extracted subimage around the center of the original image with respect to its dimensions. Figure (1) illustrates the selection of the extracted 40×40 patch from the frames of each video object. The red dotted big square illustrates one frame per video, while the smaller blue square represents the extracted 40×40 subimage. This is selected by firstly finding the position of the central(yellow) pixel and then shifting 40 rows and 40 columns before that. In this way, the red upper left corner is found, which allows the extraction of

the patch from all video frames, with respect to their dimensions. This procedure results in an image dataset of 1600 dimensions and 1395 observations (1600×1395), which are reduced to 1274 after the aforementioned filtering (1600×1274).

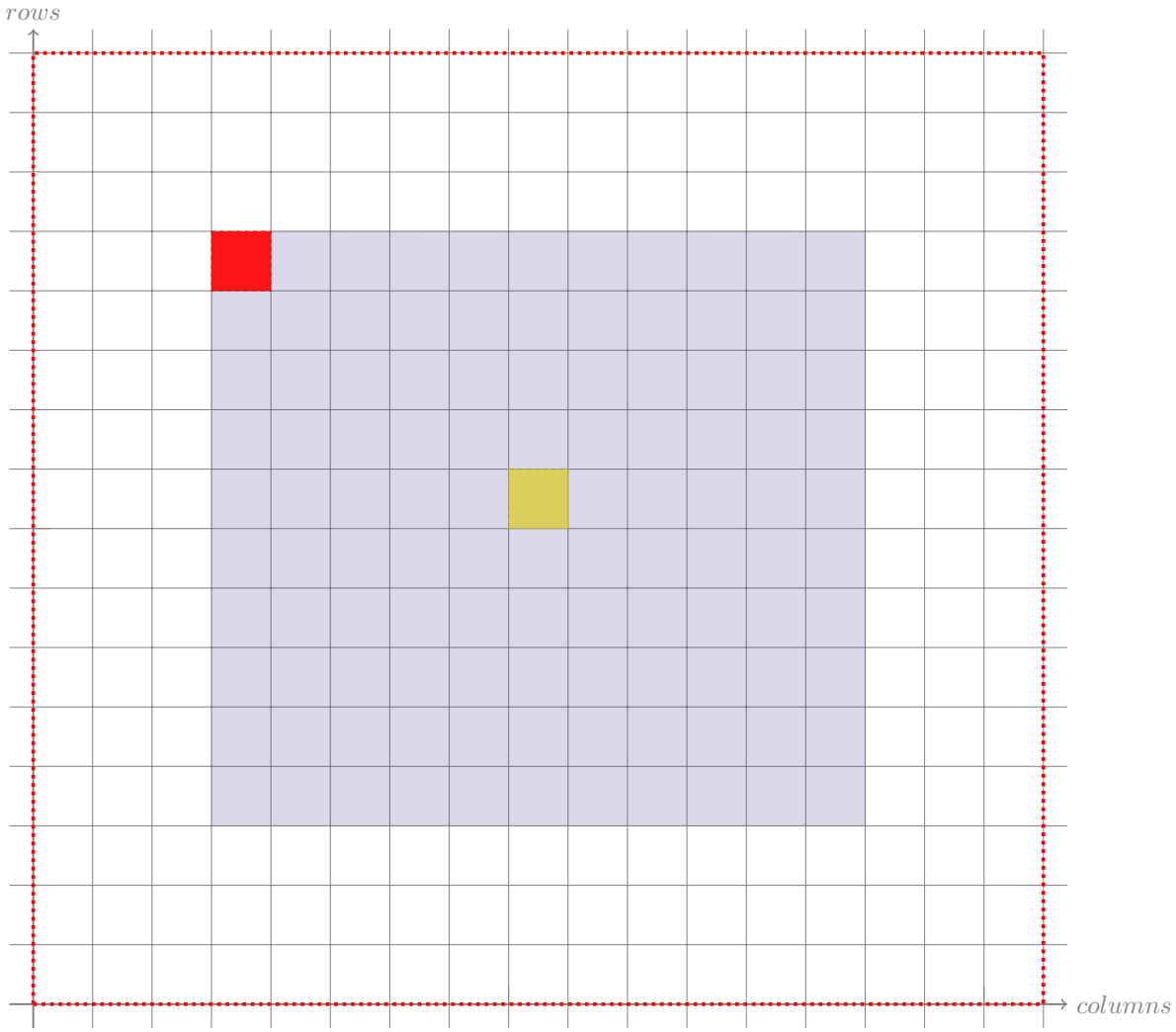
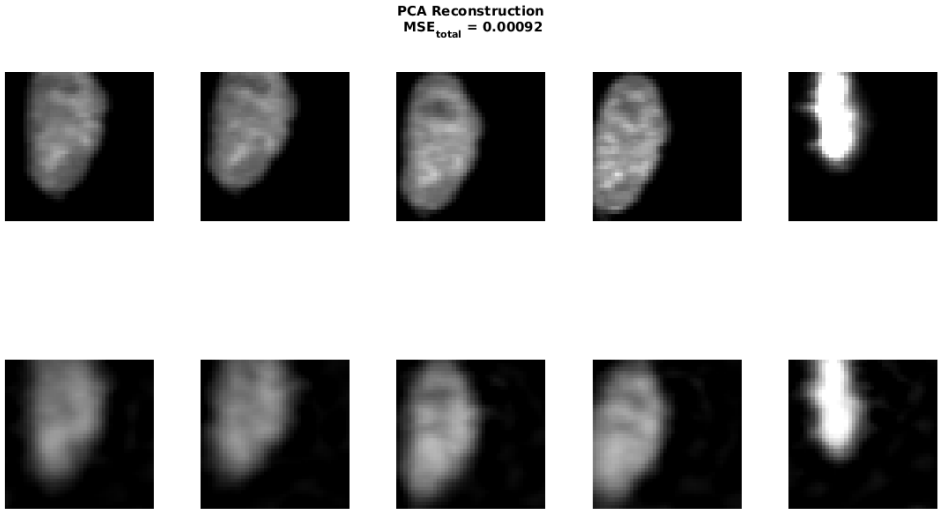
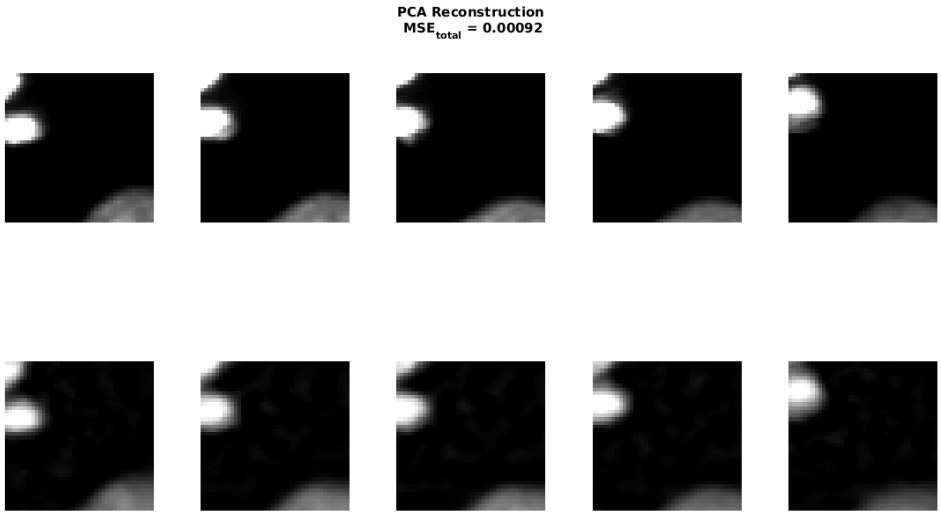


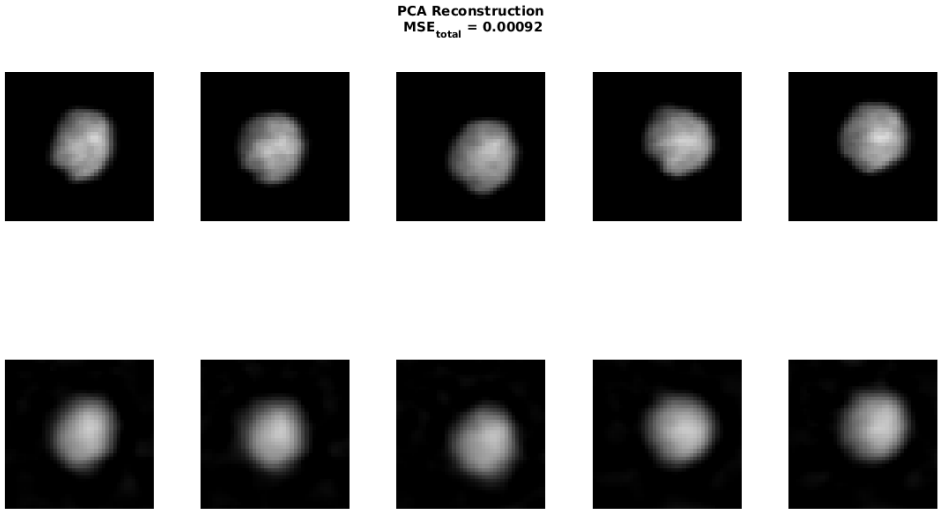
Figure 1: *Extracted subimage from a frame by dynamically positioning the upper left red corner*

1.1 Principal Component Analysis

After performing principal component analysis on the 1600-dimensional matrix of the 1274 images, it is found that the first fifty latent variables ($\mathbf{L_A = 50}$) are adequate for compression and a visually reasonable reconstruction. The corresponding total mean squared error is estimated to be $\mathbf{E_{PCA} = 0.00092}$. The following plots depict three sets of original versus reconstructed images after the compression to 50 dimensions.


PCA Reconstruction
MSE$_{total}$ = 0.00092


PCA Reconstruction
MSE$_{total}$ = 0.00092



1.2 Further compression with Bottleneck Neural Networks

In this second part of Task A, it is examined if further compression on the initially compressed inputs with plain PCA can be achieved, using deep bottleneck networks, while maintaining the same or almost the same level of total reconstruction error. Two different models of neural networks' initialization are used; *principal component analysis* (PCA) and *ordinary stacked autoencoders* (SAE), which were developed as parts of the previous assignments.

1.2.1 PCA Initialization

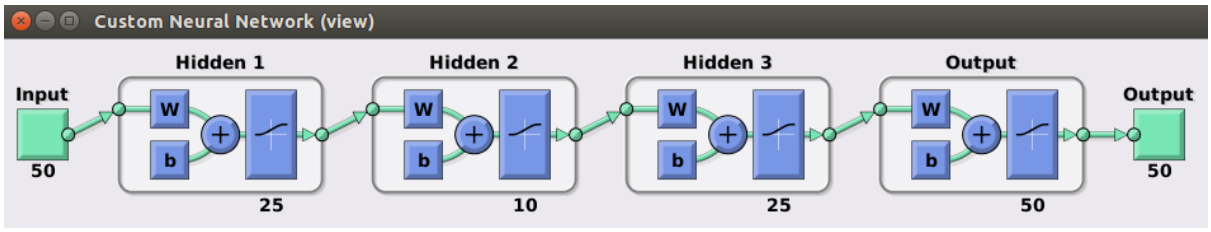
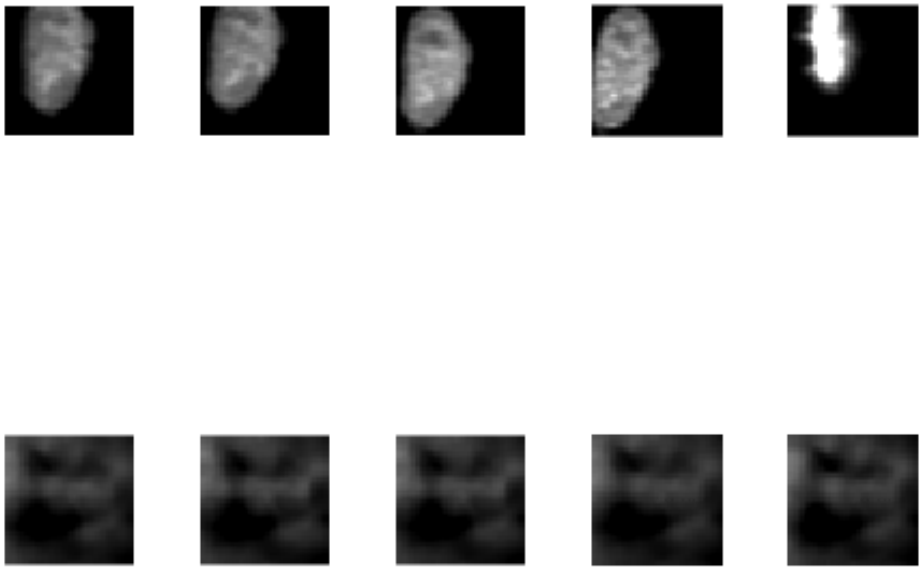
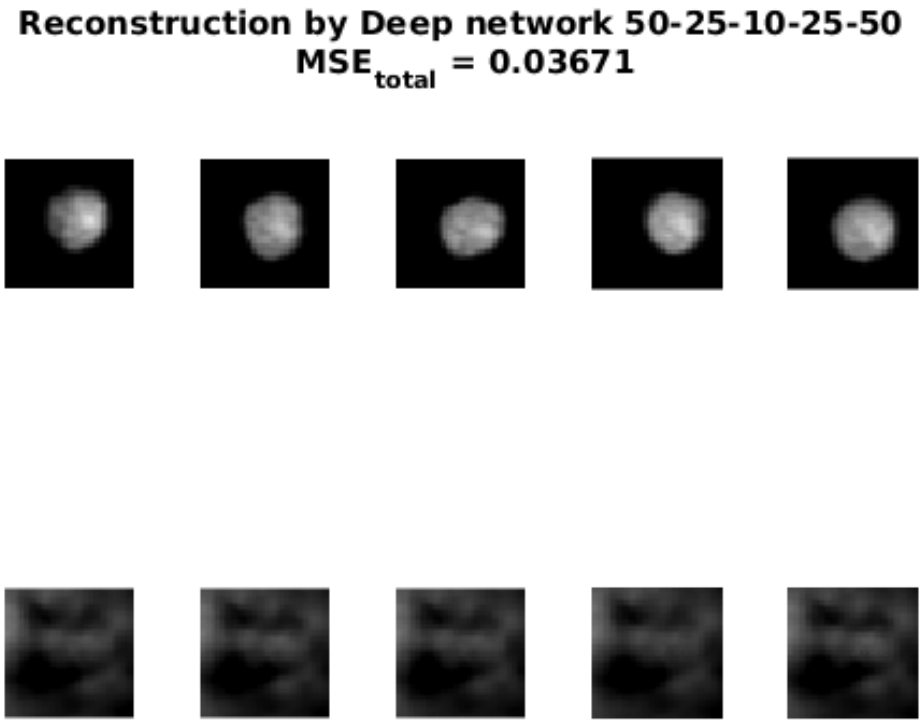


Figure 2: *Deep Neural Network 50-25-10-25-50 with PCA initialization*

Reconstruction by Deep network 50-25-10-25-50
MSE_{total} = 0.03671





1.2.2 Ordinary Stacked Autoencoder Initialization

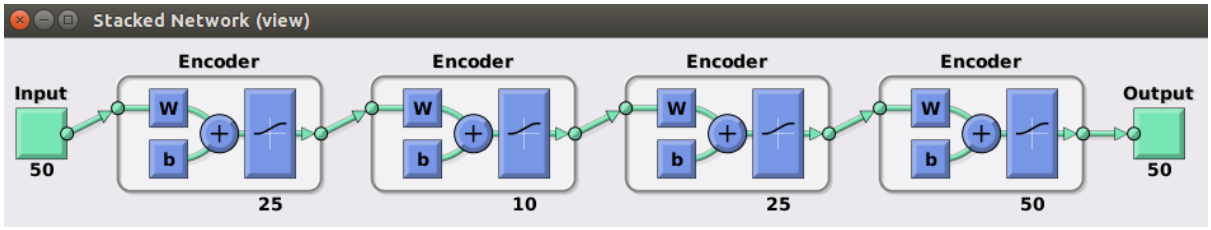
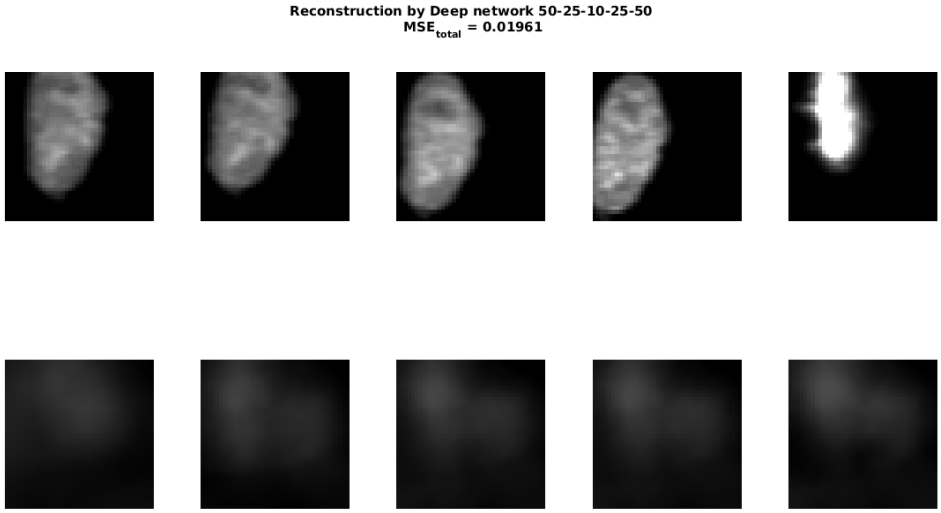
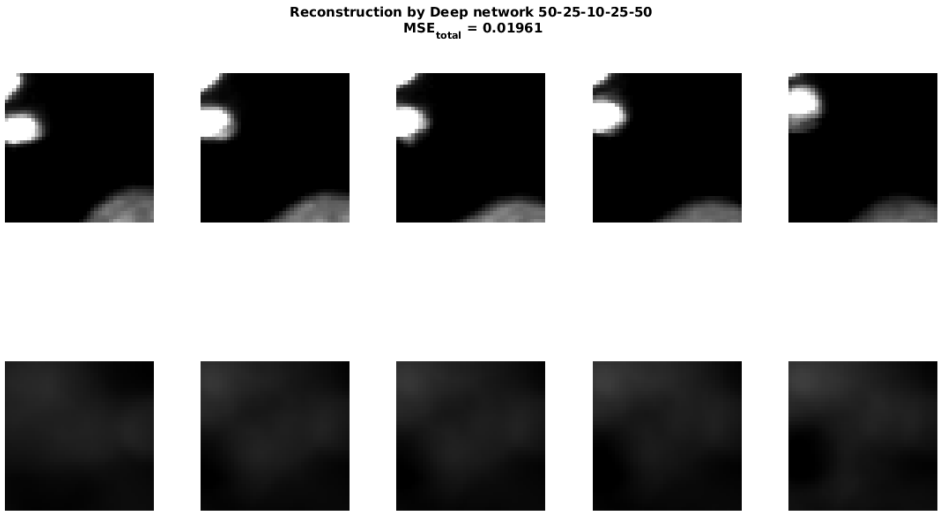
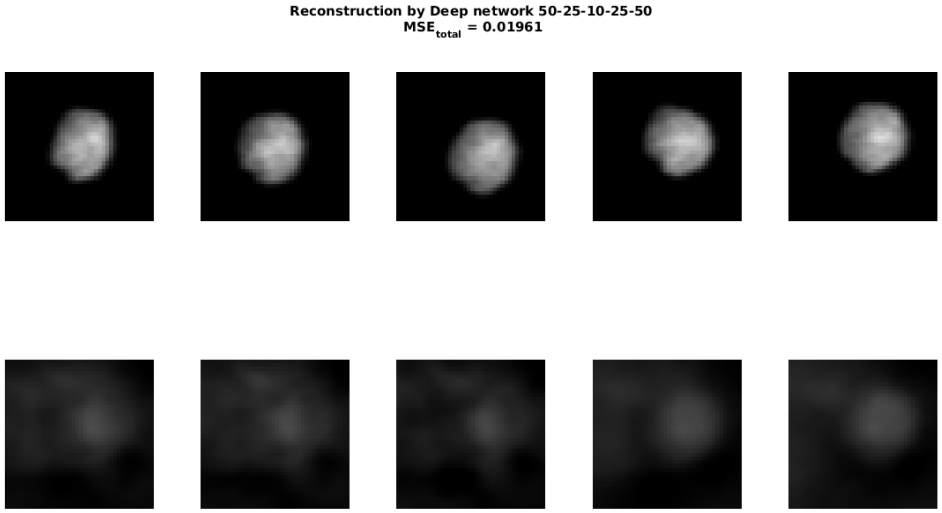


Figure 3: *Deep Neural Network 50-25-10-25-50 with ordinary stacked autoencoder initialization*


Reconstruction by Deep network 50-25-10-25-50
MSE total = 0.01961


Reconstruction by Deep network 50-25-10-25-50
MSE total = 0.01961



1.2.3 Discussion

Table (1) summarizes the results obtained by using the 50-dimensional images, after PCA compression, as the training set of two differently initialized 3-hidden layer deep neural networks:

Method		**M**ean **S**quared **E**rror
Plain PCA ($\mathbf{L = 50}$)		0.00092
Initialization	PCA	0.03671
	SAE	0.01961

Table 1: *Reconstruction MSE on the training set*

It is obvious that it cannot be achieved further compression without losing reconstruction error both in the case of PCA and SAE initialization. Other than that, the resulting reconstructions are visually intorelable, especially in the case of PCA based initialization.

1.3 Bottleneck Neural Networks with uncompressed inputs

In this last part of Task A, bottleneck neural networks with different structures are trained using the uncompressed set of images. More precisely, two different structures are tested; $\{1600 - 90 - \mathbf{50} - 90 - 1600\}$ and $\{1600 - 90 - \mathbf{35} - 90 - 1600\}$. It is important to notice that the former one has as many neurons in the second hidden layer as the $\mathbf{50}$ selected latent variables after PCA with $E_{PCA} = 0.00092$, while the latter one has less neurons. The goal is to examine if by keeping the size of the second hidden layer equal to 50 or reducing it, the resulting total reconstruction error is close to that one of plain PCA. The initialization of weights and biases of the deep neural networks is implemented by employing ordinary and denoising stacked autoencoders.

1.4 PCA Initialization

1600-90-50-90-1600

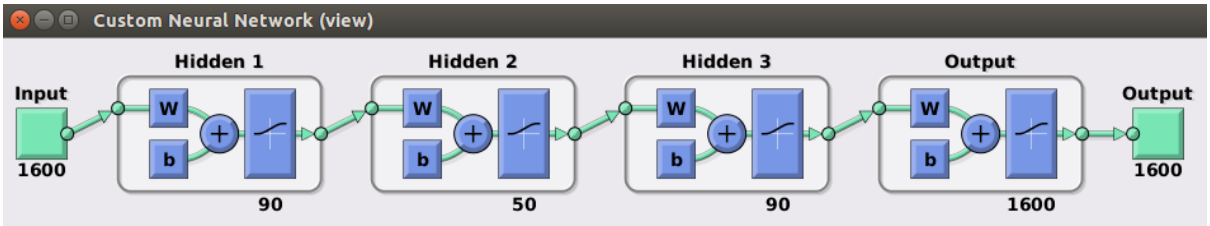
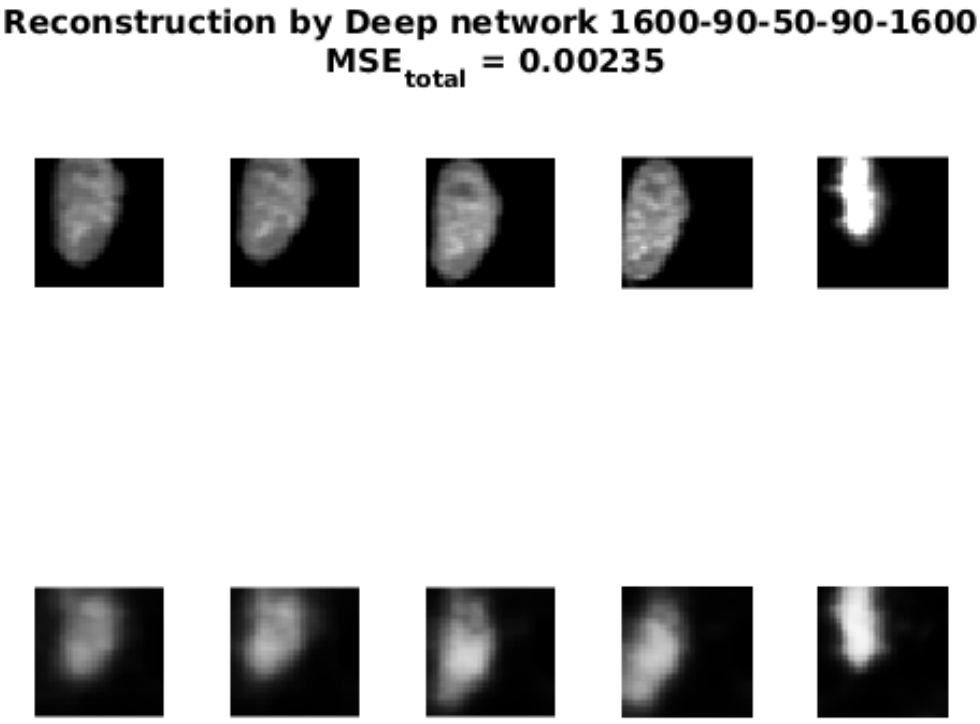
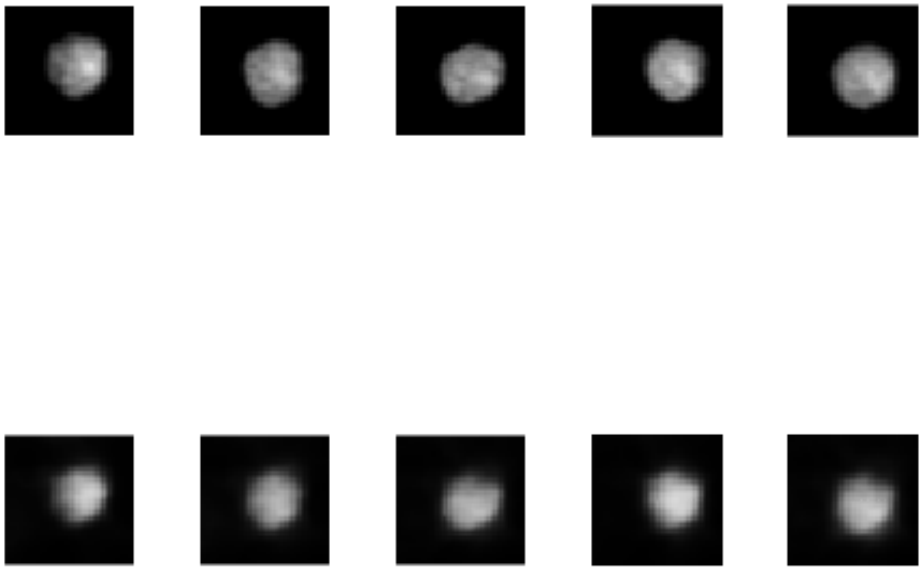


Figure 4: *Deep Neural Network 1600-90-50-90-1600 with PCA initialization*

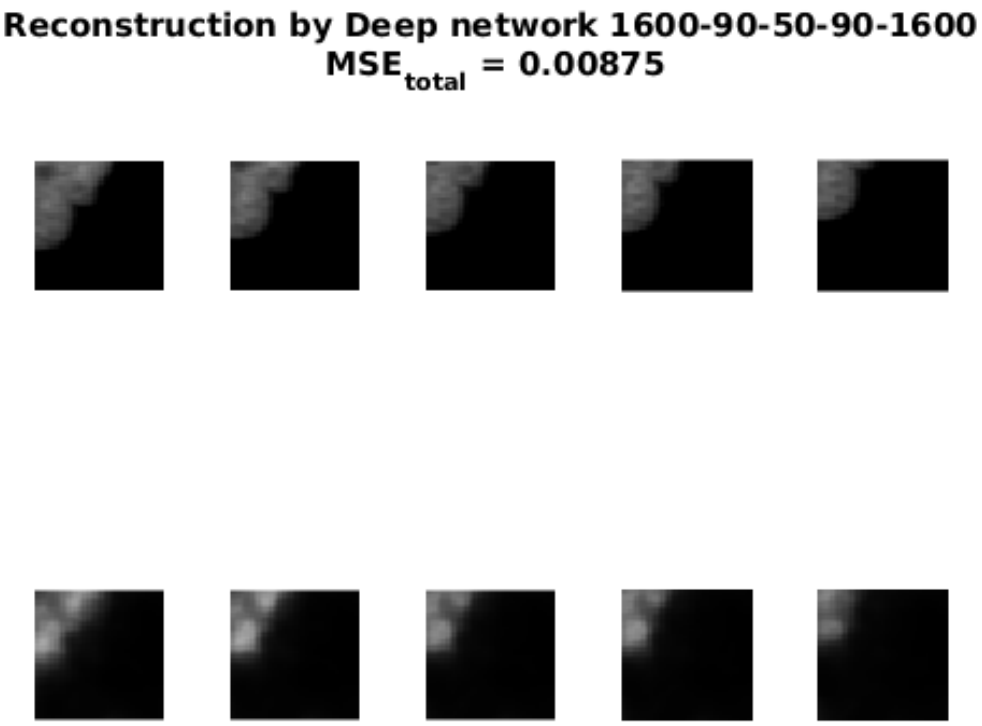
Training Dataset

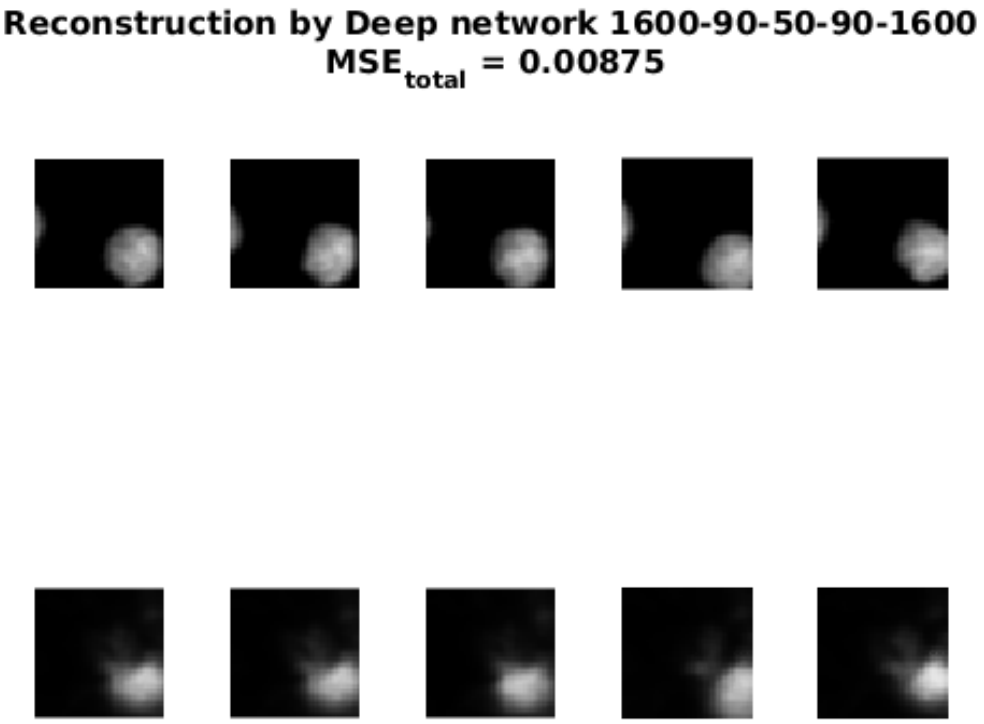


Reconstruction by Deep network 1600-90-50-90-1600
MSE_{total} = 0.00235



Test Dataset





1.4.1 Ordinary Stacked Autoencoder Initialization

1600-90-50-90-1600

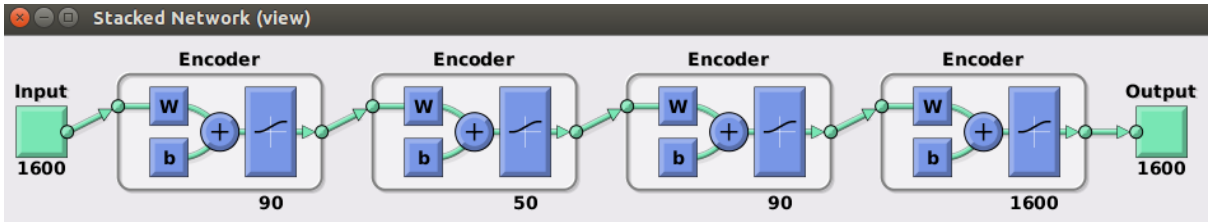
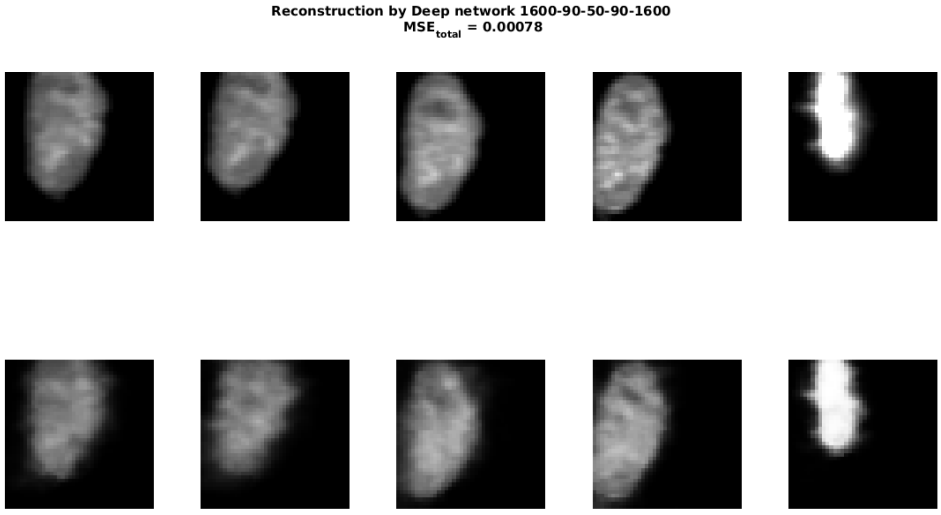
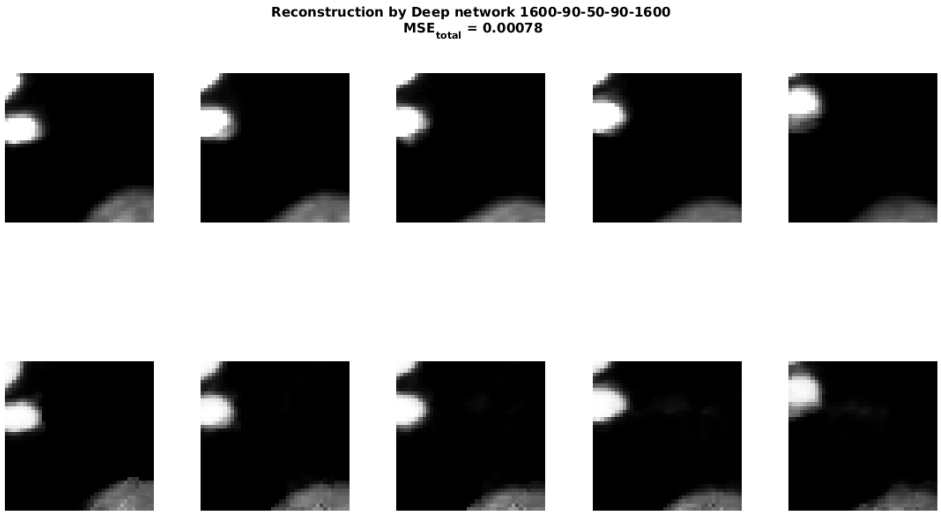
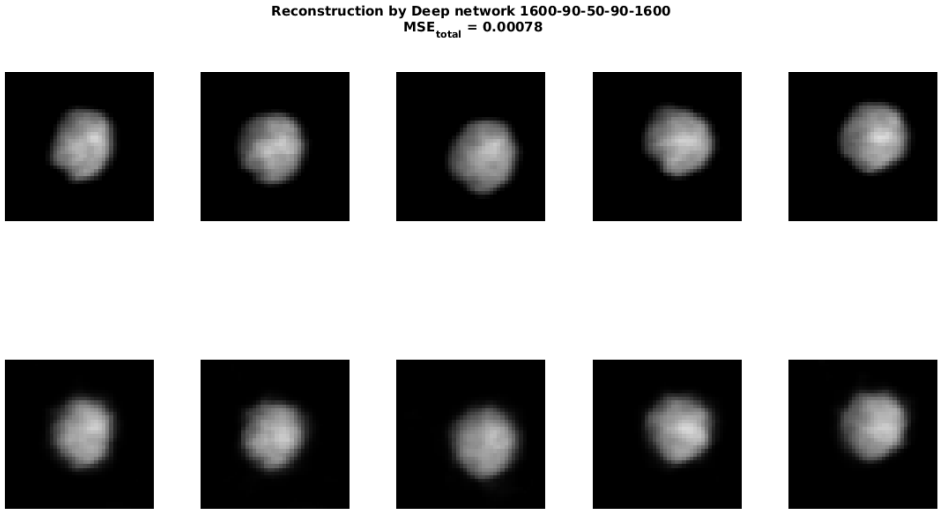


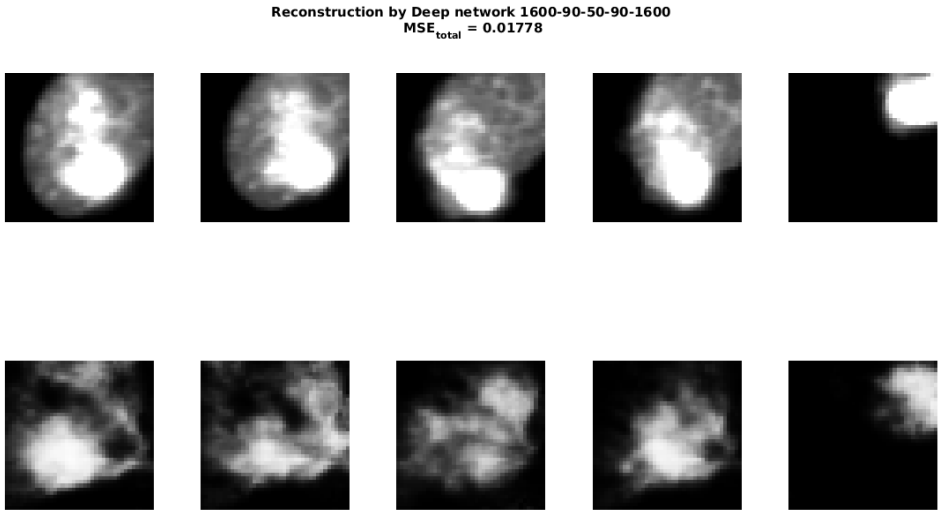
Figure 5: *Deep Neural Network 1600-90-50-90-1600 with SAE initialization*


Reconstruction by Deep network 1600-90-50-90-1600
MSE$_{total}$ = 0.00078


Reconstruction by Deep network 1600-90-50-90-1600
MSE$_{total}$ = 0.00078



It is important to evaluate the performance of the trained deep neural network on an independent test set, in order to assess the ability of the network to generalize unseen data. For this reason, I created an independent set of test examples(1600×1148) in the same way as the training examples. The only difference is that I used different coordinates for the extracted patch, so as to ensure that the training and test patches are different. As indicated by the following figure, the reconstruction on the test examples is far from being desired and the total mean reconstruction error is almost 19 times bigger than this one of the training set. Consequently, this is an indication of overfitting that leads to poor predictive performance of the network on new unseen data.



1600-90-35-90-1600

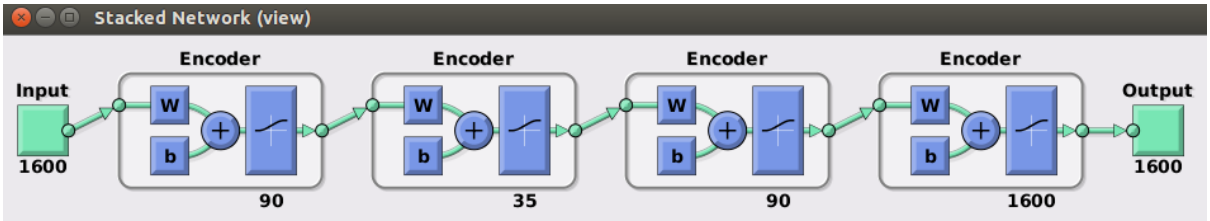
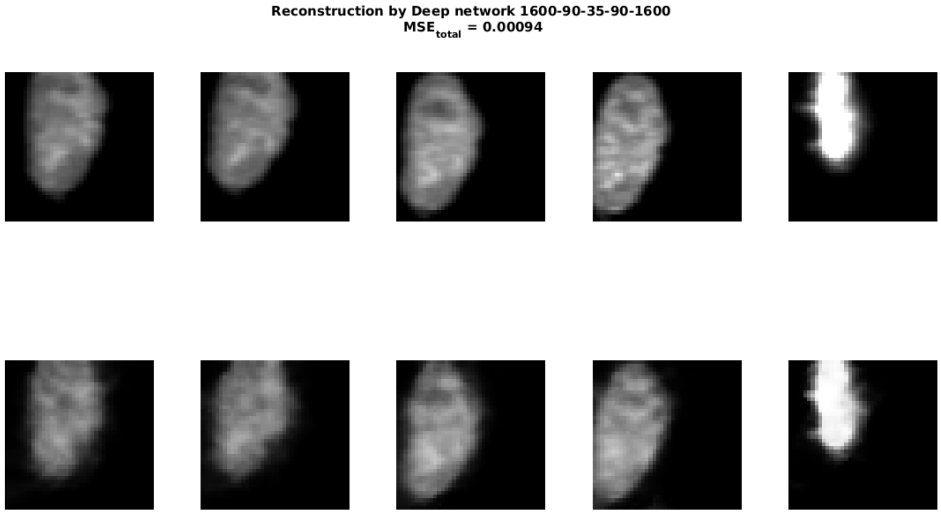
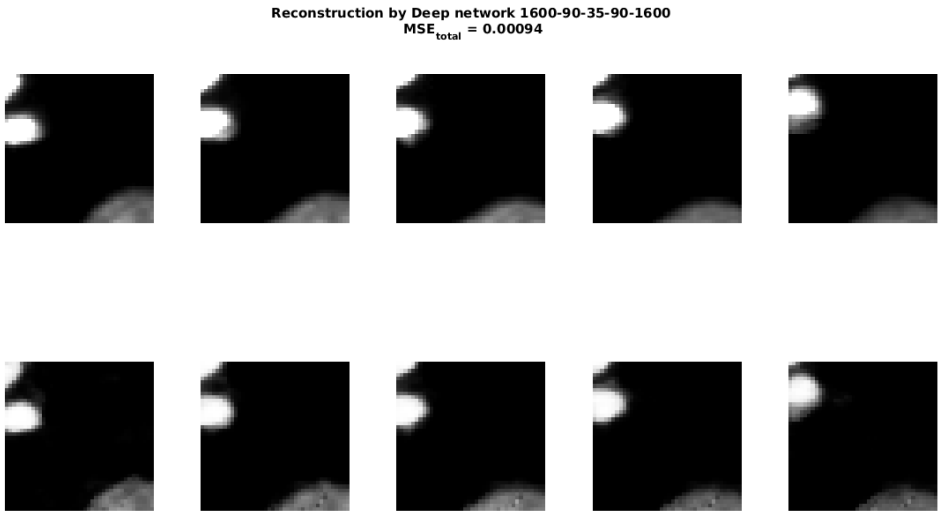
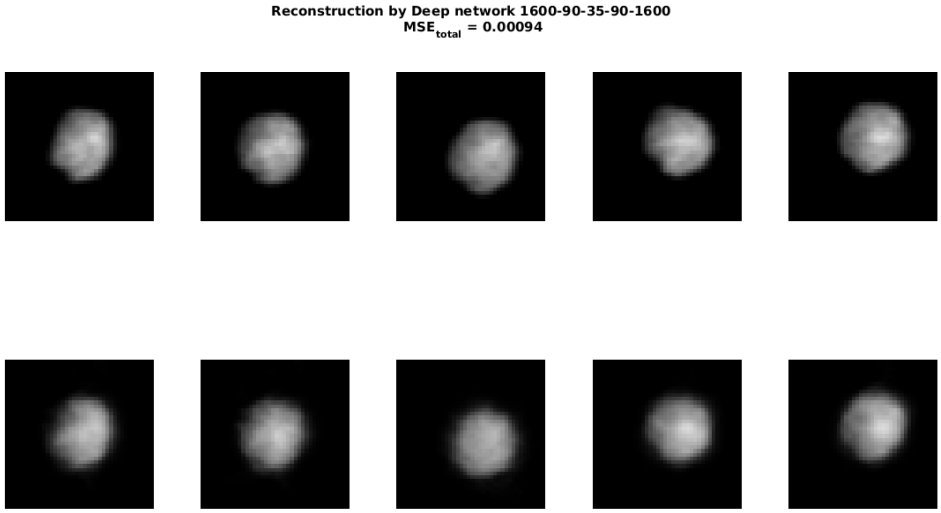


Figure 6: *Deep Neural Network 1600-90-35-90-1600 with SAE initialization*







1.4.2 Denoising Stacked Autoencoder Initialization

This type of initialization was implemented during the previous exercise (Exercise 4). In this case, the inputs of each layer are corrupted with pepper noise (density $= 0.1$) during the pre-training procedure.

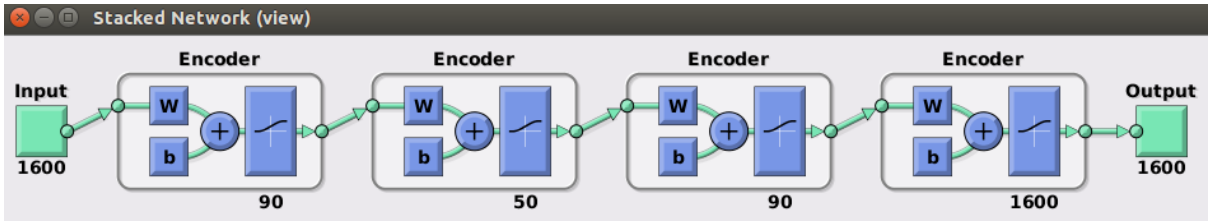
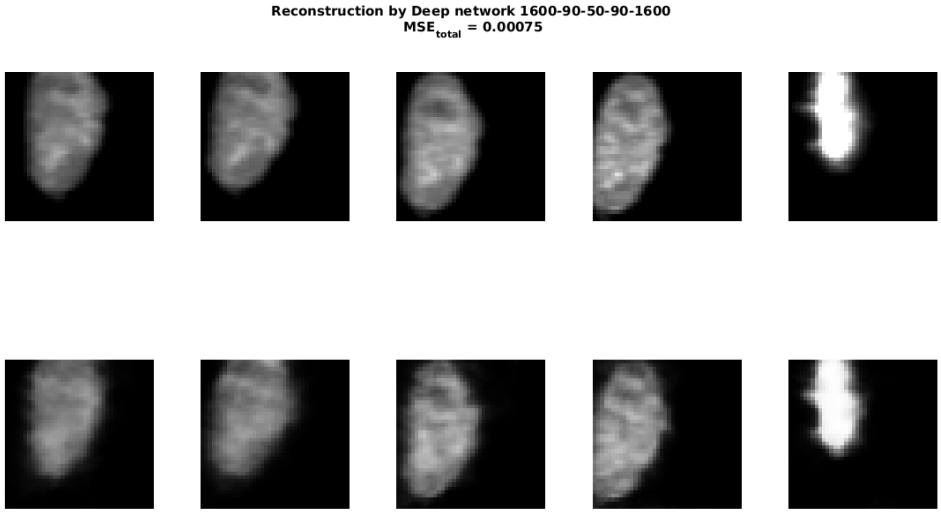
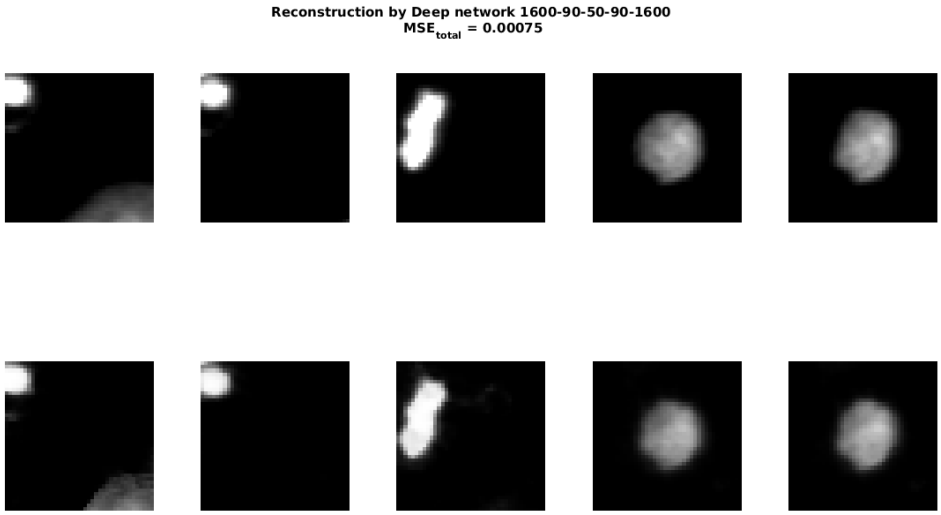
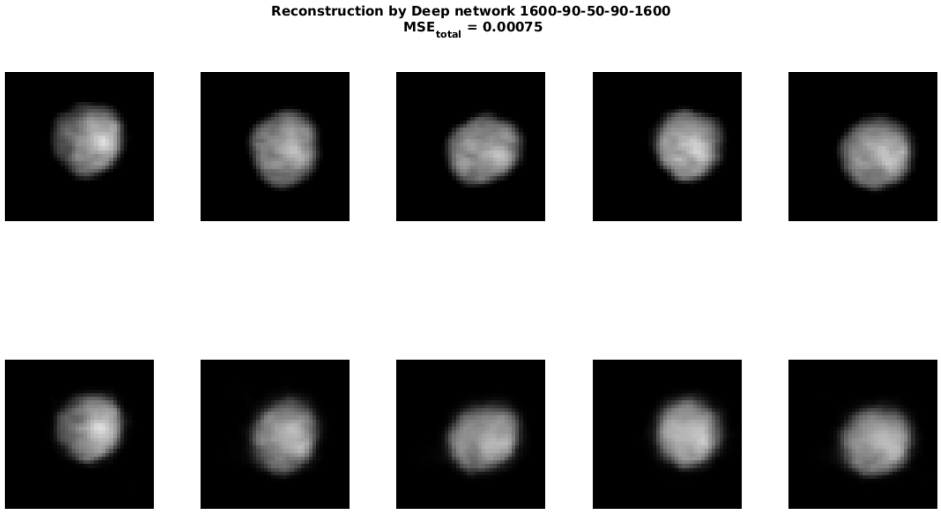


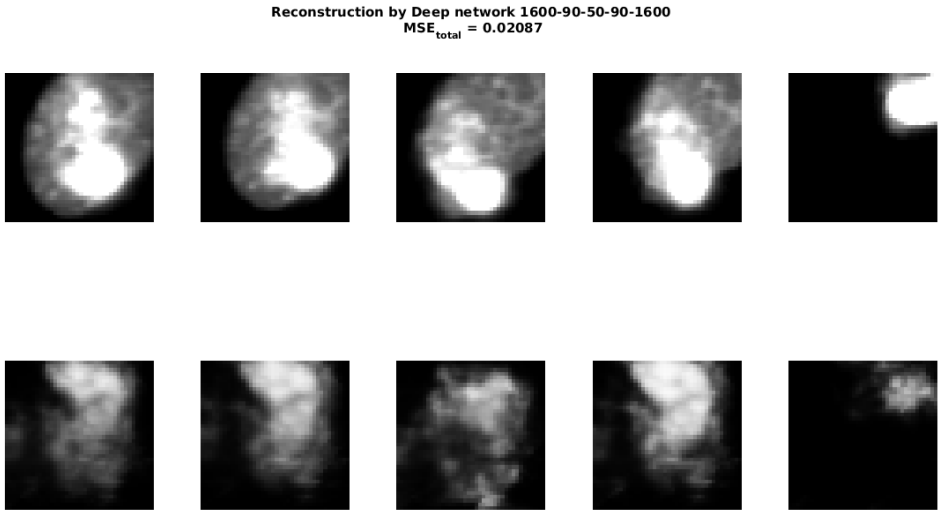
Figure 7: *Deep Neural Network 1600-90-50-90-1600 with DSAE initialization*


Reconstruction by Deep network 1600-90-50-90-1600
MSE$_{total}$ = 0.00075


Reconstruction by Deep network 1600-90-50-90-1600
MSE$_{total}$ = 0.00075



Similarly as in the case of the ordinary stacked autoencoder initialization, the performance of the fine-tuned deep neural network is evaluated on an independent test set; the same used for the ordinary stacked autoencoders. The results here verify those obtained in the case of the ordinary stacked autoencoders. The network seem to have memorized the training examples, without being able to generalize on new unseen data.



1.4.3 Discussion

Table (2) summarizes the results gained by training deep neural networks with 3 hidden layers using *ordinary* and *denoising* stacked autoencoder initialization on the uncompressed training(1600×1274) and test(1600×1148) datasets of images.

Initialization	Structure	**M**ean **S**quared **E**rror		Time (hrs : min : sec)
		Training Set	Test Set	
PCA	$1600-90-50-90-1600$	0.00235	0.00875	$00:03:19$
SAE	$1600-90-50-90-1600$	0.00078	0.01778	$00:54:10$
	$1600-90-35-90-1600$	0.00094	0.02734	$00:44:09$
DSAE	$1600-90-50-90-1600$	0.00075	0.02087	$00:57:45$

Table 2: *Reconstruction errors using deep neural networks with PCA, SAE and DSAE initialization on the uncompressed training and test time-lapse movie datasets*

Judging by the obtained results included in table (2), it is obvious that PCA based initialization outperforms SAE and DSAE initializations on the independent test set both in time requirements and performance estimates.

2 Task B

In this part, the initial 1600-dimensional dataset of 1274 images is split randomly into two equally large halves. In this way, there are two independent sets, which can serve as the training and test set, respectively. Evaluating the performance on an independent test set is essential, in order to figure out if the trained model overfits the learning examples and thus, it ends up having a poor predictive performance.

The main concept in this case is to apply PCA either on the first or second half and select the number of latent variables (L_1) that result in a total mean reconstruction error close to that one gained by performing PCA on the whole input dataset in task A. Then, the unused subset must be compressed to the L_1 dimensions and reconstructed. The goal is to get a reconstruction error, which is approximately equal both to that one of the first subset and the whole dataset.

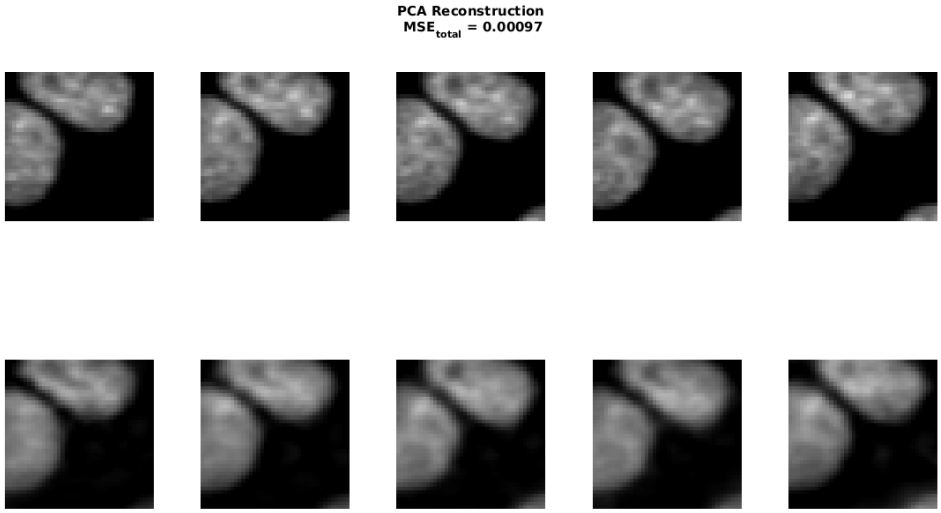


Figure 8: *PCA on the first subset* $\longmapsto L_1 = 47$, $E_{PCA1} = 0.00097$

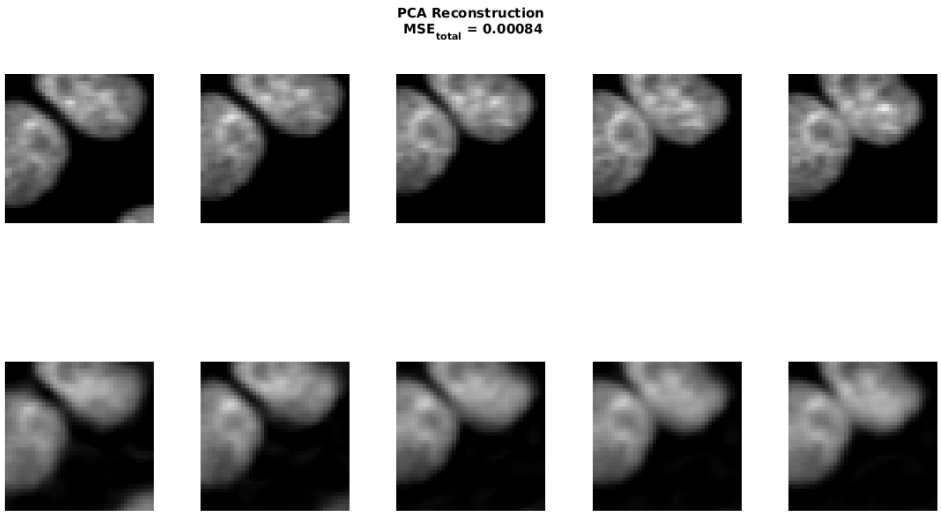


Figure 9: *PCA on the second subset using* $L_1 = 47 \longmapsto E_{PCA2} = 0.00084$

The obtained results are summarized in table (3):

Principal Component Analysis		
Dataset	Latent Variables	**M**ean **S**quared **E**rror
1600×1274 (whole)	$L = 50$	$E_{PCA} = 0.00092$
1600×637 (1^{st} half)	$L_1 = 47$	$E_{PCA1} = 0.00097$
1600×637 (2^{st} half)	$L_1 = 47$	$E_{PCA2} = 0.00084$

Table 3: *Latent variables and respective reconstruction errors on different subsets of the original dataset*

As indicated by the results, the minimum number of latent variables extracted by PCA on one half, which can also be used on the other half and yield similar errors to that one of the whole dataset is 47.

3 Task C

For this task, I used the largest public repository for high-throughput microarray experimental data; *NCBI Gene Expression Omnibus* (GEO) [3]. MATLAB R2015b provides built-in functions for retrieving all four entity types from the GEO database; GEO Platform(GPL), GEO Sample(GSM), GEO Series(GSE) and curated GEO DataSet(GDS).

I retrieved human RNA expression data of glioblastoma stem-like (GS) cell lines, with GEO Series id *GSE23806* [4]. The corresponding expression matrix constists of 92 samples and $54,675$ genes. In other words, the input dataset is a 92-dimensional matrix of $54,675$ vectors of genes (92×54675).

In first place, PCA is applied on this mRNA expression dataset with the intention of compression to fewer L_C dimensions, while maintaining, at the same time, the average relative reconstruction error less than 0.05. Then, the resulting compressed set is used to train bottleneck networks with PCA and SAE initialization. In other words, as with the time-lapse movie dataset, it is examined if further compression can be achieved but not in the expense of higher reconstruction error.

The obtained results are summarized in table (4):

Method		**M**ean **R**elative **E**rror
Plain PCA ($\mathbf{L_C = 34}$)		0.0496
Initialization	Structure	**M**ean **R**elative **E**rror
PCA	$34-17-6-17-34$	0.08756
SAE	$34-17-6-17-34$	0.12431

Table 4: *Relative reconstruction errors after PCA and further compression on the GSE23806 expression matrix*

As indicated by the results, the original 92-dimensional expression matrix can be compressed to 34 dimensions with an average relative reconstruction error less than 0.05. However, this compressed set of expression profiles cannot be further compressed using PCA and SAE initialization of deep neural

networks with 3 hidden layers, without losing reconstruction error. Actually, in this case, the average relative error ends up being almost 3 times bigger.

References

[1] "MathWorks Neural Network Toolbox." http://se.mathworks.com/products/neural-network/. Accessed: December 4, 2015.

[2] B. Neumann *et al.*, "Phenotypic profiling of the human genome by time-lapse microscopy reveals cell division genes," *Nature*, vol. 464, pp. 721–727, Apr 2010.

[3] "Gene Expression Omnibus." http://www.ncbi.nlm.nih.gov/geo/. Accessed: December 4, 2015.

[4] "Gene Expression Omnibus." http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE23806. Accessed: December 4, 2015.