University of Padua, Department of Information Engineering

- Neural Network and Deep Learning -

Autoencoders for digit reconstruction

Giulia Bressan, ID 1206752

A.Y. 2019/2020

1 Introduction

The assignment of the fourth homework was to train a neural network as an autoencoder on the MNIST dataset. The goal was to find the right hyperparameters of the network, in order to test the reconstruction capability of the autoencoder in standard MNIST images and in others, corrupted with various levels of noise and occlusions.
The work is structured as follows:

1. Loading of the dataset and creation of the corrupted test set,
2. Tuning of the hyperparameters of the autoencoder using k-fold cross validation with grid search,
3. Training of the autoencoder with the optimal parameters,
4. Test of reconstruction capability of the autoencoder.

2 Train and test dataset

The dataset used for the training of the network is the standard MNIST dataset. It is made of images of handwritten digits from 0 to 9, with a dimension of 28x28 pixels. The training set has a size of 60 000 samples, while the test set of 10 000.
During the training of the network the samples used are the original ones, without any noise or

occlusion corruption. In the test set, instead, the samples were corrupted. The corruption was done in two ways:

- Adding gaussian noise to the original images, sampled from a gaussian distribution with zero mean and standard deviation of 0.1, 0,05 and 0,03;
- Adding occlusions of different shapes, ranging from 0.02 to 0.33 the area of the original image and with range of aspect ratio of the erased area from 0.3 to 3.3.

The final test set was made of original images, occluded images and images corrupted with noise. The probability with which the images were corrupted in different ways is shown in Table 2.1. Since the application of the corruption mechanisms was following a probability rule, it happened that some images were both occluded and corrupted with noise.

Type of corruption	**Probability**
No corruption	0.33
Occlusion	0.33
Gaussian noise, std = 0.1	0.11
Gaussian noise, std = 0.05	0.11
Gaussian noise, std = 0.03	0.11

Table 2.1: Different type of corruption applied to the images and probability of application.

The following images show the different samples generated.

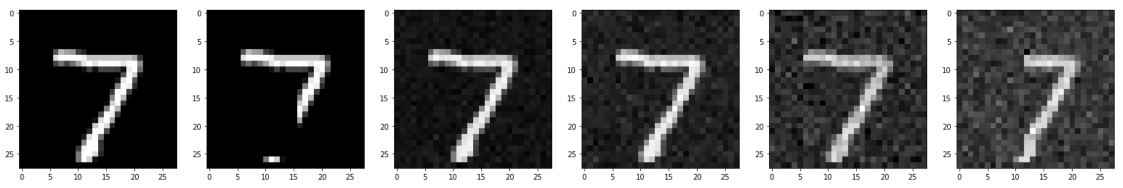

Figure 2.1: Examples of images from the test dataset. From the left to the right: no corruption, occlusion, corruption with gaussian noise with standard deviation of 0.1, 0.05, 0.03, superimposition of occlusion and gaussian noise.

3 Model

The model of the autoencoder used for the assignment is the one proposed during the lab session.
It is composed of two main parts: the first is the encoding scheme, followed by the decoder.
The composition of the two parts is as follows:

- **ENCODER**, composed of:
  - **convolutional part**: made of three 2D-convolutional layers, with increasing output channel dimension of 8, 16 and 32, each followed by a *ReLU* activation function;
  - **linear part**: made of a first linear layer with output dimension of 64, followed by a *ReLU* activation function, and a second linear layer with output dimension corresponding to the dimension of the encoded space (tested with different values);
- **DECODER**, composed of:
  - **linear part**: made of two linear layers with output dimension of 64 and 32, each followed by a *ReLU* activation function;

- **convolutional part**: three transposed 2D-convolutional layers, with decreasing input channel dimension of 32, 16 and 8, each followed by a *ReLU* activation function, except for the last one.

As a loss function for the network, the *Mean Squared Error* was used, and as an optimizer the *Adam* optimizer with fixed weight decay of 0.00001.
The learning rate of the optimizer was chosen using a cross-validation procedure, as well as the batch size for the training.
The values of the parameters tested with the k-fold cross-validation were the following:

Parameter	**Possible values**
Learning rate	0.001 - 0.0001 - 0.00001
Batch size	256 -512 - 1024

Table 3.1: Set of hyperparameters used during the cross-validation with grid search.

With a training of 10 epochs and a division of 3 folds, the optimal values found were a learning rate of 0.001 and a batch size of 256, which were then used for the final training of the network. To test the behaviour of the network with different dimension of the encoded space, also this parameter was tested during the cross validation with values of 2, 4, 6, 8, even if the final training was done with different values as well.

4 TRAINING

The training of the network was done for 163 epochs, with an early stopping mechanism. If the loss for the test set (corrupted, as shown in Section 2) did not improve for 20 consecutive epochs, the training stopped. The final training loss was of 0.014781.
Then, also different values of the encoded space were tested (2, 4, 6 and 8), and the results suggested a behaviour in line with the results obtained with the cross-validation. In particular, a smaller dimension employed a bigger number of epochs to get similar losses as the ones obtained with higher dimensions, but never reaching the same results. This is due to the fact that with an higher dimensional space, the compression of the images is less severe, so the reconstruction is easier and more accurate, giving a smaller test loss after the decoding. In Table 4.1 a comparison of the results for the different values is shown.

Encoded space dimension	**Final test loss**	**Number of epochs**
2	0.043248	157
4	0.031387	174
6	0.027985	208
8	0.021381	163

Table 4.1: Loss and number of epochs of training for the different encoded space dimensions tested.

In Figure 4.2 the behaviour of the losses for the three cases of dimension 2, 4, and 6 is shown, and we can see that, comparing with the case of dimension 8, the losses do not reach the same minimum value.
In the following section the results refer only to the case of the encoded space dimension of 8.

5 Results

I tested the trained network both with the original dataset and with the corrupted version of it (as shown in Section 2). I did also some tests with the dataset corrupted only with occlusions and only with the different noise levels.
Table 5.1 shows the obtained results.

Dataset	**Final test loss**
Corrupted	0.021381
Non corrupted	0.014735
Only occlusion	0.029718
Only Gaussian noise, std = 0.1	0.025539
Only Gaussian noise, std = 0.05	0.017463
Only Gaussian noise, std = 0.03	0.015723

Table 5.1: Loss for the different dataset tested.

As we can see, the best result is obviously obtained with the non-corrupted dataset, since the training is done with a non-corrupted dataset too.
Results show also that with a low noise level the reconstruction capabilities of the network are similar to the ones of the dataset without corruption. As we increase the standard deviation of the noise to 0.1 the loss increases, but still the reconstructed image is very similar to the original one, without noise.
The worst result is obtained with the dataset with occlusions only. But even if the loss seems high, the quality of the reconstructed images is sometimes very impressive. There are some challenging cases in which the digit is mostly occluded and even the human eye is not able to distinguish the number, so the reconstruction fails, but the results are overall very good.

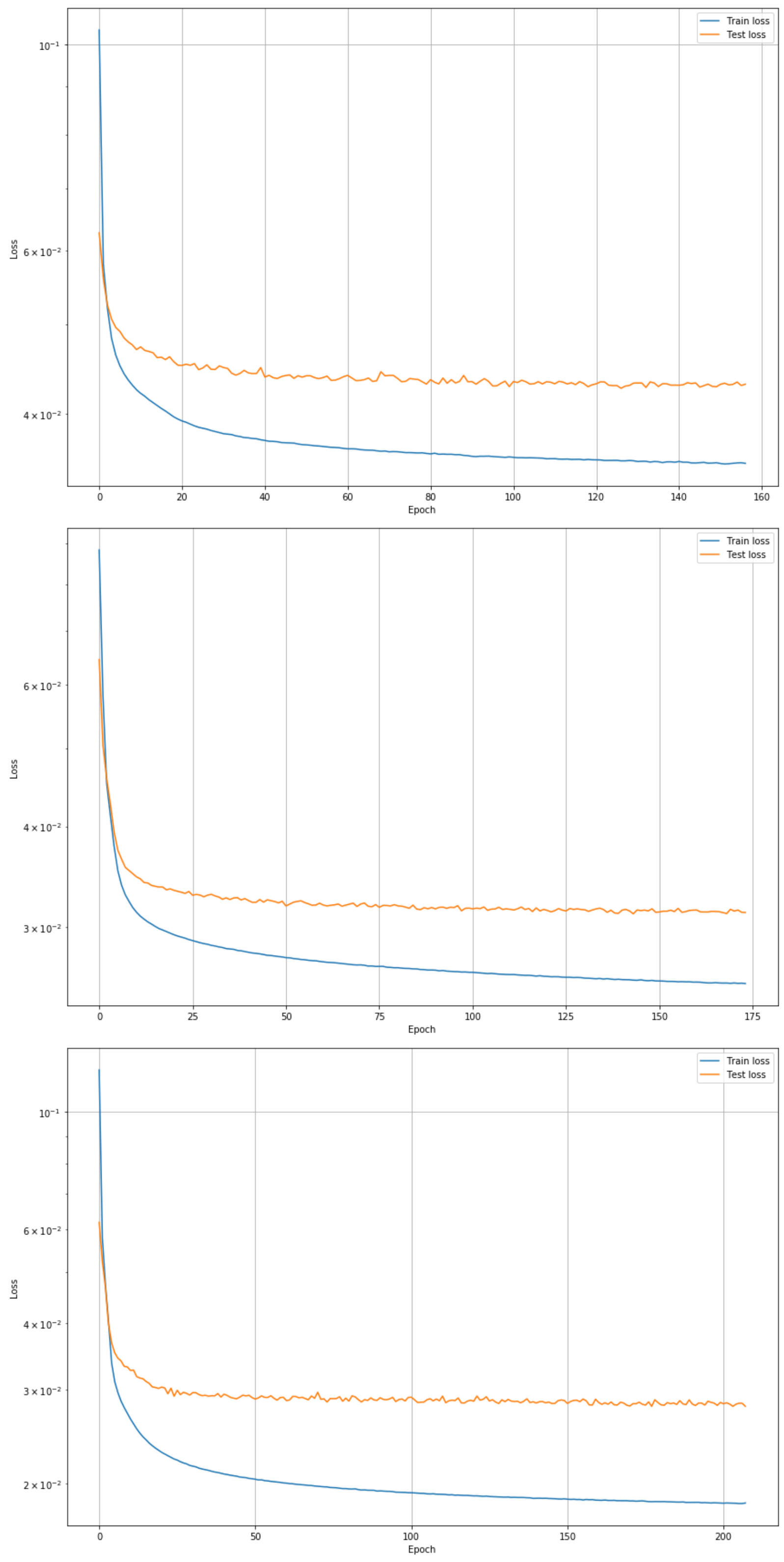

Figure 4.1: Train and test loss behaviour for the different encoded space dimensions tested (from top to bottom6: dimension equal to 2, 4 and 6).

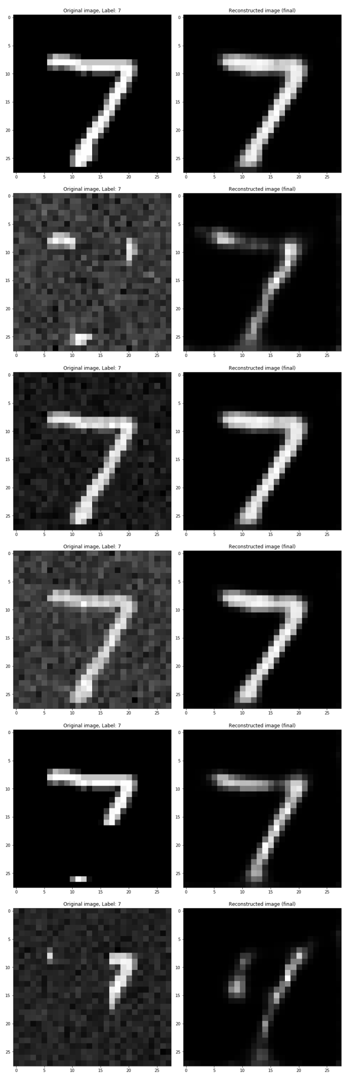

Figure 5.1: Examples of original vs reconstructed images, with different kind of corruption. We can see that the second one is heavily corrupted, but the reconstruction is still good, while for the last one the occlusion is too spread and the reconstruction is not as accurate as in the other cases.

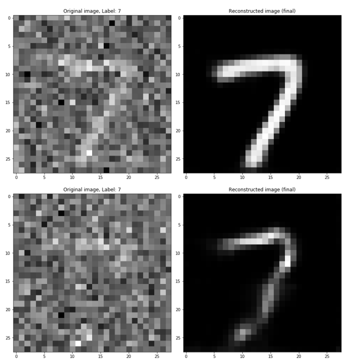

Figure 5.2: In this example the original images have a noise level higher than in the other examples of the test dataset (std = 0.5) and still the reconstruction is quite good.