HW. 4 NN and Deep Learning

Federico Bottaro

February 15, 2020

Abstract

In this document we are going to explore the field of the encoding using neural networks. We use the MNIST dataset to perform some study on the behavior of the encoding.

1 Project development

This document is divided in two main parts. The first one concerns the study of the parameters that influence the performance of the network. In particular we use a fixed (given) network structure to explore the best learning rate to use with the choosed optimizer that is an Adam optimizer. In this context we also apply a weight decay that we fix to 1×10^{-5} to reduce the risk of overfitting and to make the training more stable. Once that we find the best learning rate we move to study the peculiar feature of the encoder that is the dimension of the encoding. The second part of the document evaluate the model that we find out from the previous study in different ways:

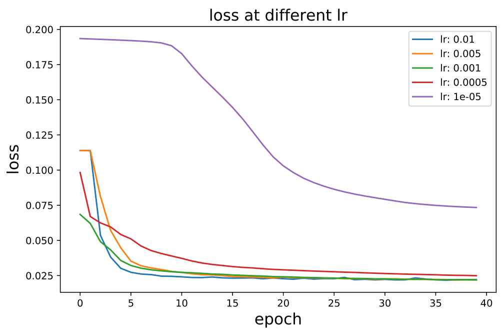
- testing the algorithm on the MNIST.mat file with original immage
- Evaluate the impact of random noise that is added to the original images.
- Evaluate the impact of the occlusion of some parts of the original images.

2 Research of the best model

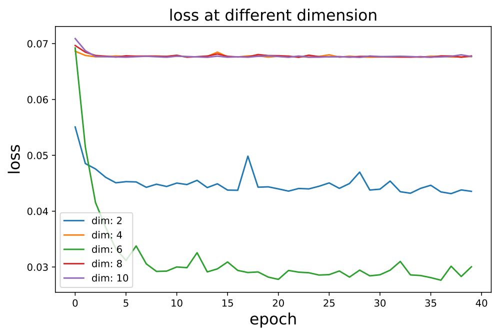
We can move to the first topic of the document. Using the given structure of the network we perform a grid search on the learning rate using the values $[1 \times 10^{-2}, 5 \times 10^{-3}, 1 \times 10^{-3}, 5 \times 10^{-4}, 1 \times 10^{-5}]$. Figure 1a shows the behavior of the loss on a evaluation set during the training. In the table bellow we have the value of the MSE evaluated on the test set.

lr	MSE
0.01	0.0221
0.005	0.0218
0.001	0.0219
0.0005	0.0249
0.00001	0.0733

We choose a learning rate of 0.005 that is the value with the smaller MSE and we can move to evaluate which dimension of the features space of the compression performs better. The values of the dimension tested are: $[2, 4, 6, 8, 10]$. Figure 1b shows the behavior of the loss on a evaluation set during the training. As we can see with a dimension of 6 we get the best results in term of loss function. So from now on we keep fixed this structure for the neural network.



(A) *Loss of a validation set for different lr*



(B) *Loss of a validation set for different dimensions*

Figure 1: *Some numbers contained in the MNIST dataset*

Once that the model is choosed we perform a final training and Figure 2 shows the loss function during the training for both the training set and the evaluation set.

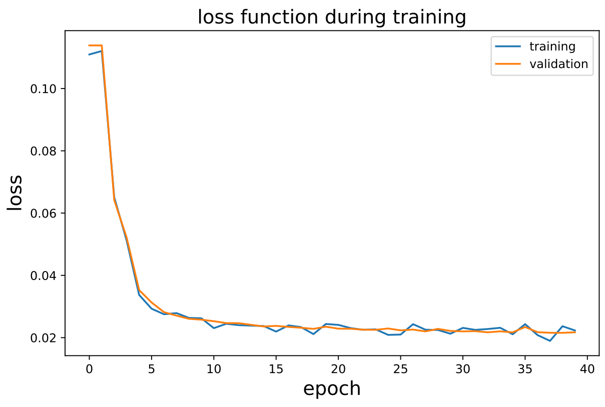
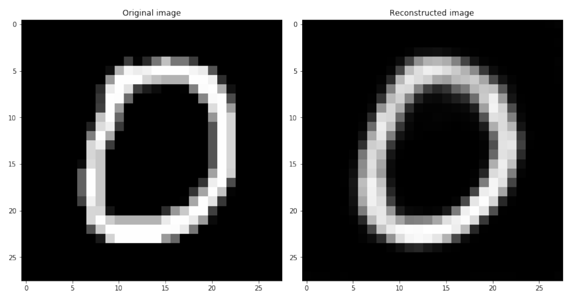
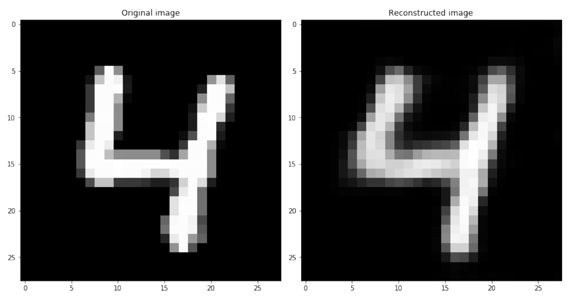


Figure 2: *Loss function for the final training*

Figure 3 shows some example keeped from a test set.



(A)



(B)

Figure 3: *Some examples of encoding and decoding using the trained network. For each pairs the first number is the original one while the second number is the reconstructed by the network.*

Another interesting feature that can be studied is how the encoded space of the numbers present in the MNIST dataset can be visualized. We have to remember that we are using a dimension of 6 so figure 4 show just a projection. As we can see there is a little bit of confusion due to the complexity of the network nevertheless we can recognize that for example the blue points (representing the number zero) are grouped on the bottom left of the plot and we can say the same for the pink points in the left.

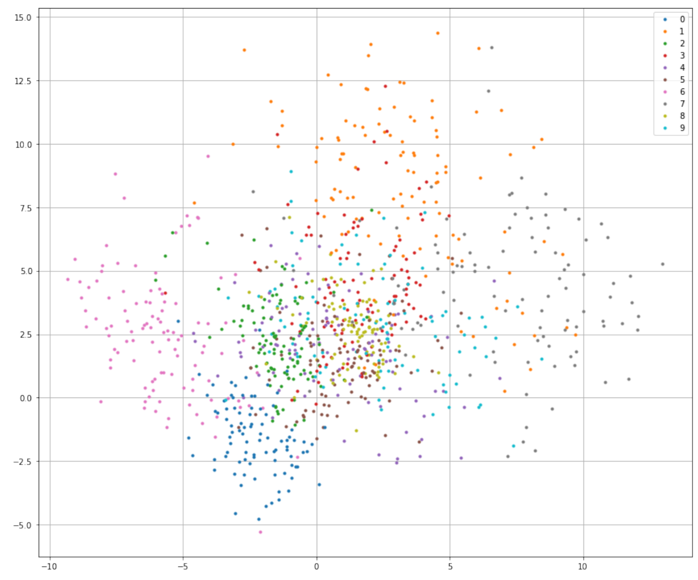


Figure 4: *Weights of the encoding*

3 Results

We can now move to the second part of the document that concerns the analysis of the model in a test dataset. For this scope we use the MNIST dataset provided as file *.mat*. Remember that during the training we used the MNIST included in torchvision. Before starting with the analysis we have to perform some preprocessing on the data infact the numbers in *MNIST.mat* are rotated and flipped top bottom. We train our network with standard oriented numbers so some change are needed.

Using the model discussed above we get an error of:

$$\mathrm{MSE} = 0.021806583$$

Also the robustness of the autoencoder was tested. Two different ideas was developped:

- Corrupting the original images adding some Gaussian random noise with different level of noise.
- Corrupting the original images deleting some part of the images; also in this case we are going to explore different level of noise.

For both the approach we evaluate the MSE on the MNIST.mat at each level of noise. For the Gaussian noise we produce for each image in the dataset an image of the same dimensions with values picked up from a normal distribution with mean 0 and with variable variance, in particular we choose ten value between 1×10^{-3} and 1. Figure 5 shows how the noise level impacts on the performances. We can have also a visual example in figure 6. The same think evaluated for the gaussian noise are showed in figure 7 and 8 with reference to occlusion of some part of the images. For make the occlusion in practice we decide to put all the values of some raws of a given tensor (that represents the image) to 0. The power of the occlusion is the fraction of raws that we deleted (from 0 to 1).

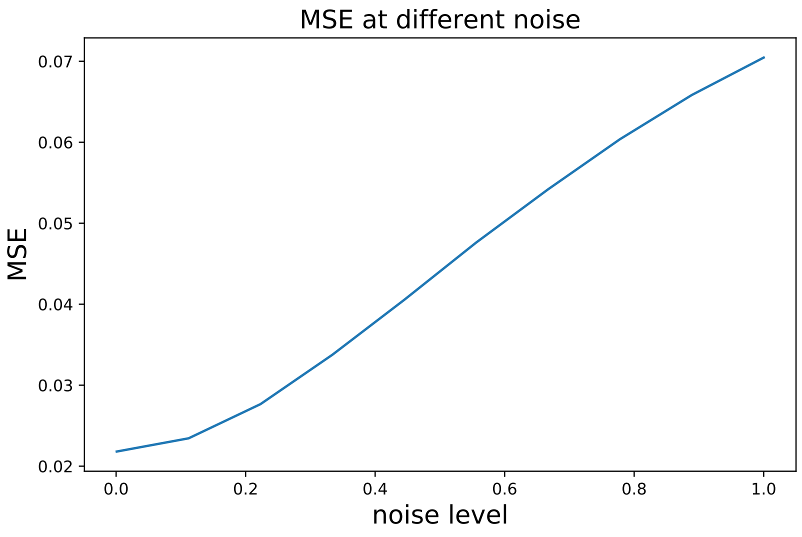
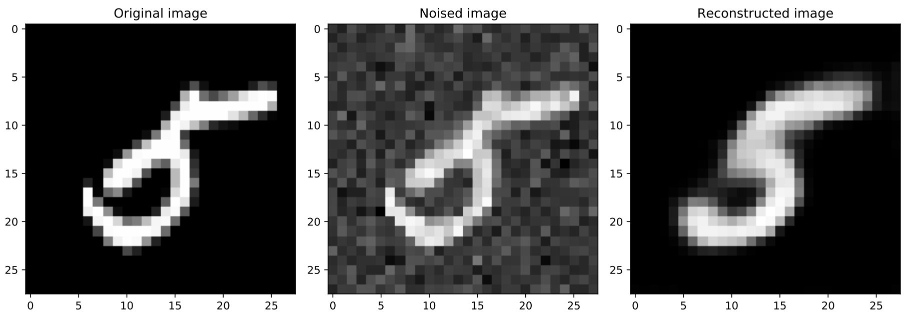
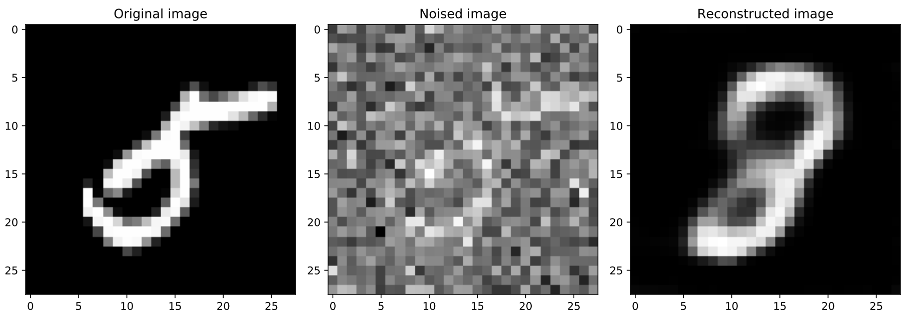


Figure 5: *MSE as function of Gaussian noise level*



(A)



(B)

Figure 6: *Some examples of how the network works with Gaussian noise*

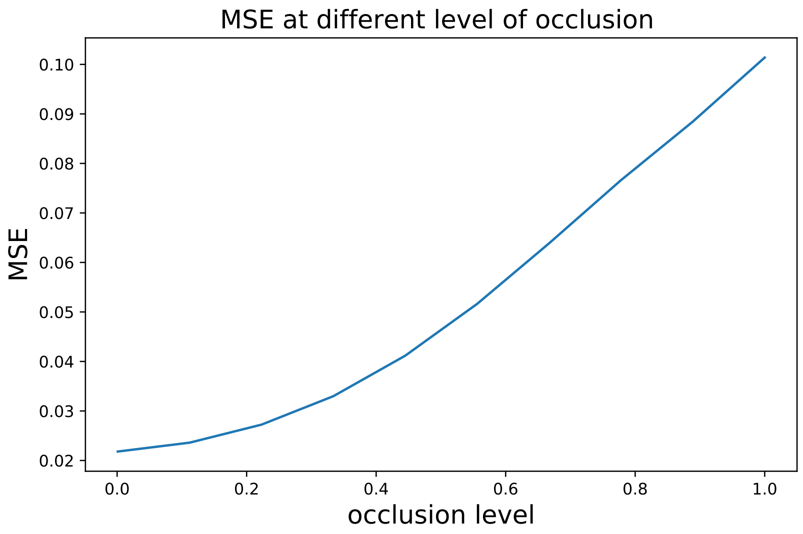


Figure 7: *MSE as function of occlusion level*

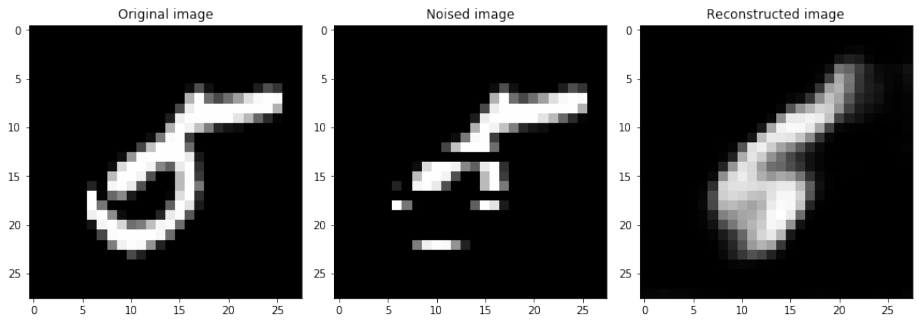


Figure 8: *Example of how the network works with occlusion*

4 Conclusion

In this homework some features of the autoencoding are explored. We have reached a model for prediction that seems to work well and is also robust to some noise (if the level goes to up we have a great error).