Unsupervised Deep Learning

Edoardo Antonaci 1234431
edoardo.antonaci@studenti.unipd.it

Abstract

In this work we explore the unsupervised deep learning world starting with two common convolutional architectures: the Autoencoder and Generative Adversarial Network(GAN) . Using the train images from MNIST dataset (without labels) we reconstruct and generate new images starting from noise images or only pure random noise. Adding a new layer on the Encoder architecture we switch from an unsupervised task into supervised one(like homework 1) and so we can do some comparisons between the two training behaviours both in term of speed and accuracy.

1 Introduction

The main task for this homework consists to generate new images or reconstruct noised one training two different models on hand-digit MNIST dataset. The architectures proved are the convolutional autoencoder and GAN . In both of them we find two convolutional networks that work togheter but in two different ways : in the Autoencoder we have a sort of "cooperative training" between Encoder and Decoder ,on the other hand,during training Gan we observe a "competition" between Generator and Discriminator.

2 Methods

2.1 Autoencoder

This model contains two different networks : Encoder and Decoder. The Encoder consists in one convolutional layer followed by two linear layers; RELU is choosen as activation function in order to nullify negative inputs and send only the positives ones. Reached the "encoded space" the network "switchs" on Decoder part where we try to reconstruct initial images. Here we substitute conv2D with transpose one and before output we apply the Sigmoid activation function in order to limit output values and help the correct convergence. Different hyperparameters are tested and in particular ,to find the best configuration, we use "grid search" and "cross validation" functions. During tests we change the dimension of encoded space,Adam's learning rate and weights decay taking fixed the numbers of epochs ,"patience time", the loss function and the optimizer. As "patience time" we mean the needful period of time used to check if average test-loss goes down during epoch training.

So using this strategy we reach the minimum error if we set:

Dim encoded space	Epochs	Learning rate	Optimizer	Weight decay
32	20	0.003	Adam	10^{-5}

Table 1: Optimal parameters Autoencoder architecture.

2.2 GAN

GAN architecture is composed by two network : generator and discriminator. This model works in this way : the discriminator's task is to distinguish real images from "fake" ones and ,on the contrary, generator's task try to fool the discriminator generating images ,more and more real, such that discriminator isn't able to understand which one comes from dataset. In other words,at the beginning, we pass only random noise to generator,it makes an image and then this "fake" image and one image draws from train dataset are passed as input to discriminator. This is an example of min-max game and it can be shortly described by the following formula :

$$\min_{G}\max_{D} E_{x\sim p_{data}}[logD(x)] + E_{z\sim p_{noise}}[log(1 - D(G(z))]$$

Both in generator and in discriminator network we use Batch Normalization layer in order to pass "white" input ,that is with 0 mean and 1 variance, to the following layer. To improve convergence the training dataset was normalized.

The best results are achieved using this configuration:

Epochs	G Learning rate	D Learning rate	Optimizer	Batch size
100	0.0005	0.0005	Adam	128

Table 2: Optimal parameters GAN architecture.

3 Results

3.1 Autoencoder

Once the best configuration is found we train the model over more epochs and we observe that,around 50 epochs, test error doesn't decrease (figure 1) :

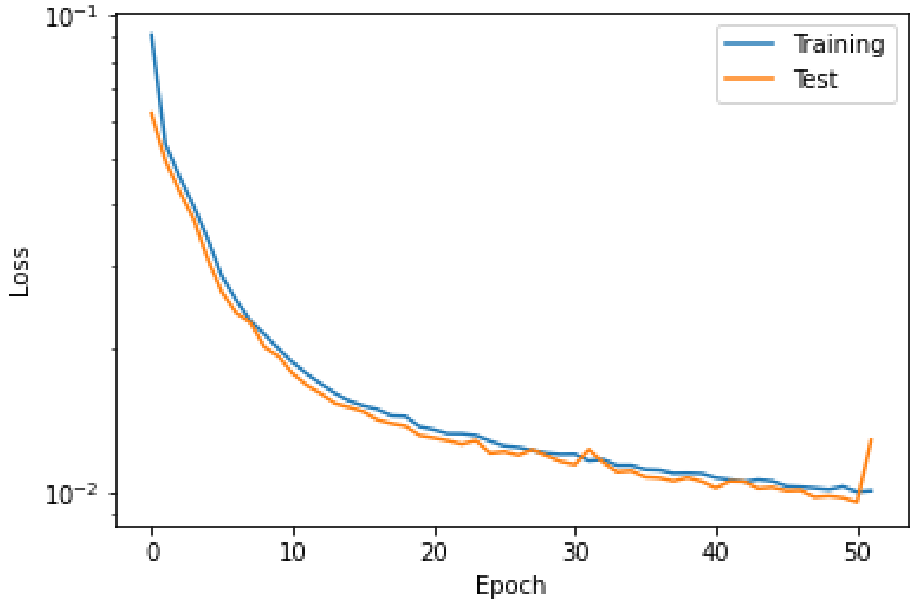


Figure 1: Best configuration training

In figure 2 we show the reconstruction process :at top the starting image and then its changes,two-epochs at a time, till the last one that corresponds to the last epoch. This is done because the changes becomes very imperceptible quite soon.

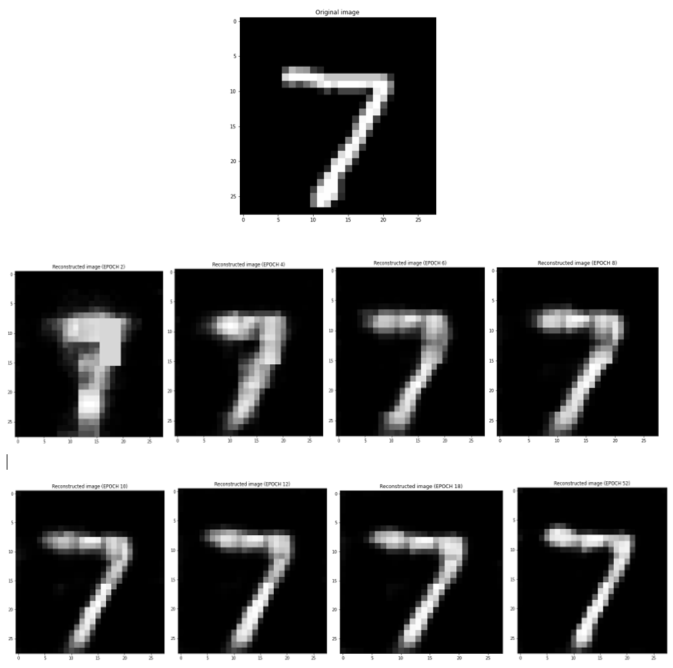


Figure 2: reconstruction images

It follows the application of the model on images with some diffuse random noise or with cropped little portions.

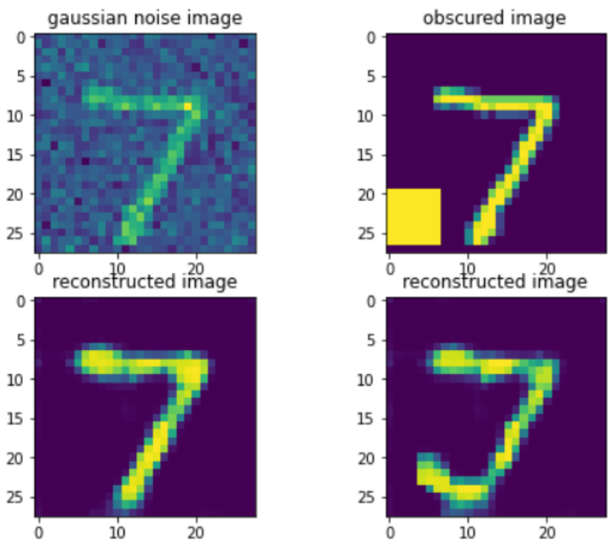


Figure 3: obscured gaussian images

Finally adding a new layer to encoder network and giving also the respective labels we pass from unsupervised training to supervised one and we can compare classification accuracy and learning speed with results achieved in homework 1

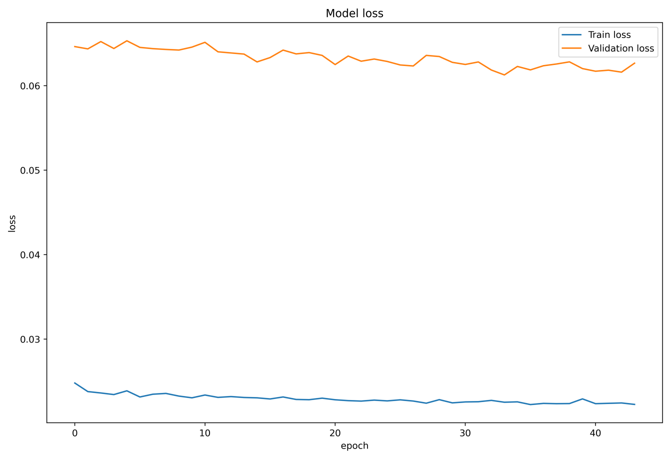
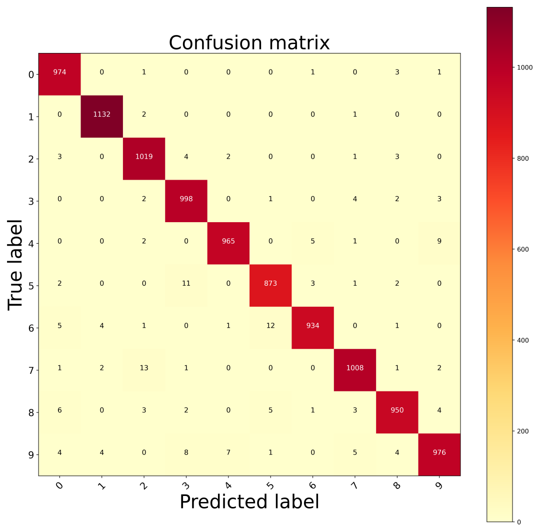


Figure 4: training in supervised learning

As shown above using the encoder + classification layer for supervised task we discover that training is very facilitated and loss results are already good at first epoch.



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	974	0	1	0	0	0	1	0	3	1
1	0	1132	2	0	0	0	0	1	0	0
2	3	0	1019	4	2	0	0	1	3	0
3	0	0	2	998	0	1	0	4	2	3
4	0	0	2	0	965	0	5	1	0	9
5	2	0	0	11	0	873	3	1	2	0
6	5	4	1	0	1	12	934	0	1	0
7	1	2	13	1	0	0	0	1008	1	2
8	6	0	3	2	0	5	1	3	950	4
9	4	4	0	8	7	1	0	5	4	976

Figure 5: Heatmap

3.2 GAN

In this section we show examples of the new samples generated by G-network during GAN training and we see how they become more and more realistic.

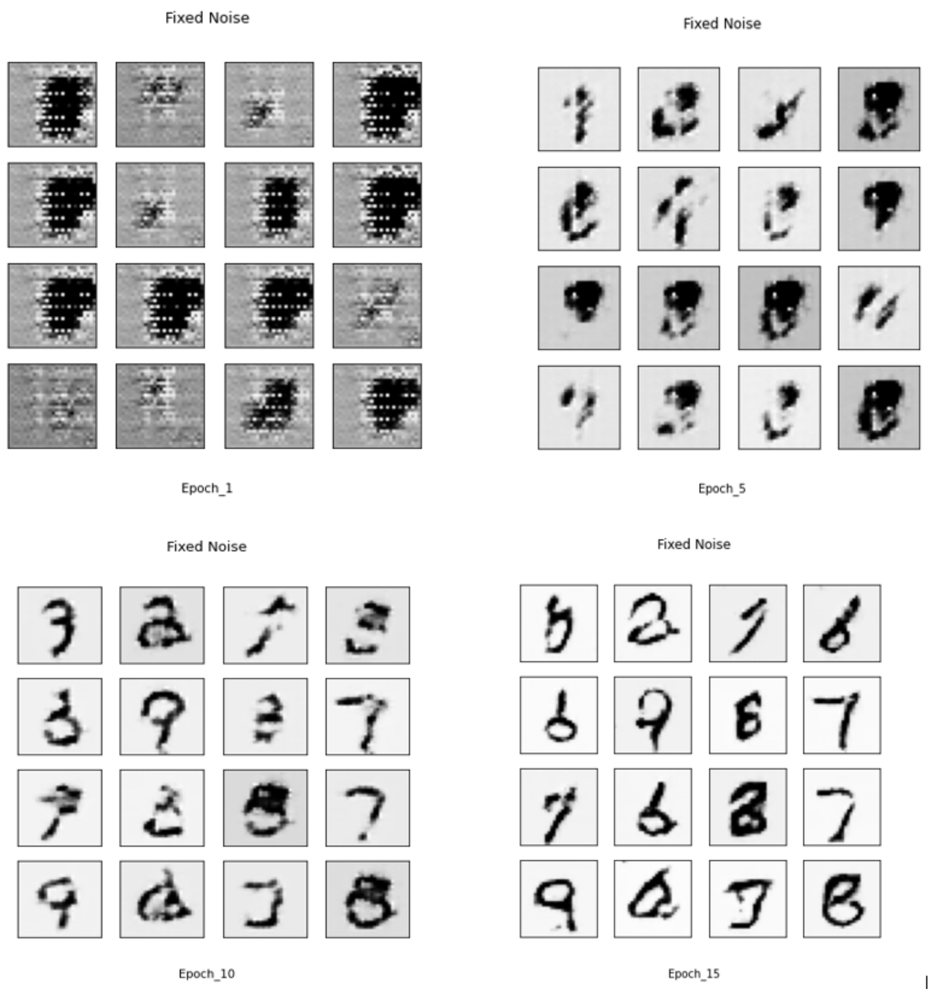


Figure 6: 1-5-10-15 epochs

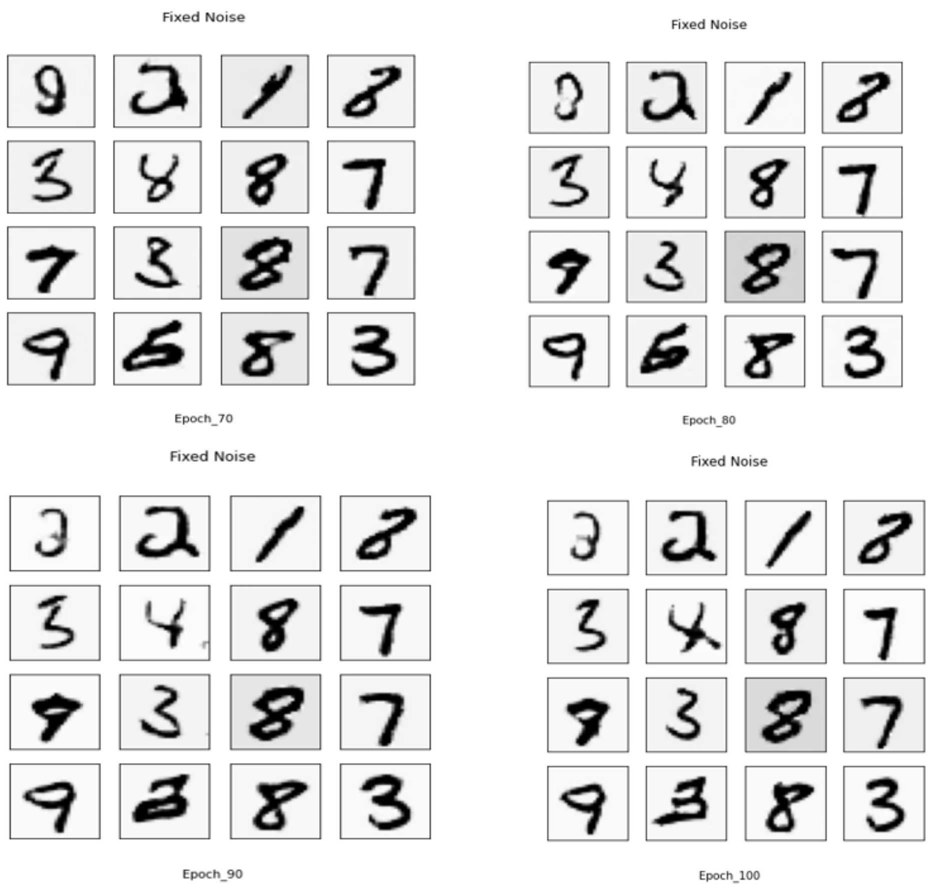


Figure 7: 70-80-90-100 epochs

4 Appendix

4.1 Autoencoder

We report also the analysis of the encoded space in figure 8 and a tentative to make a new sample giving random integer numbers. The figure 9 shows that trained decoder is not able to generate good patterns.

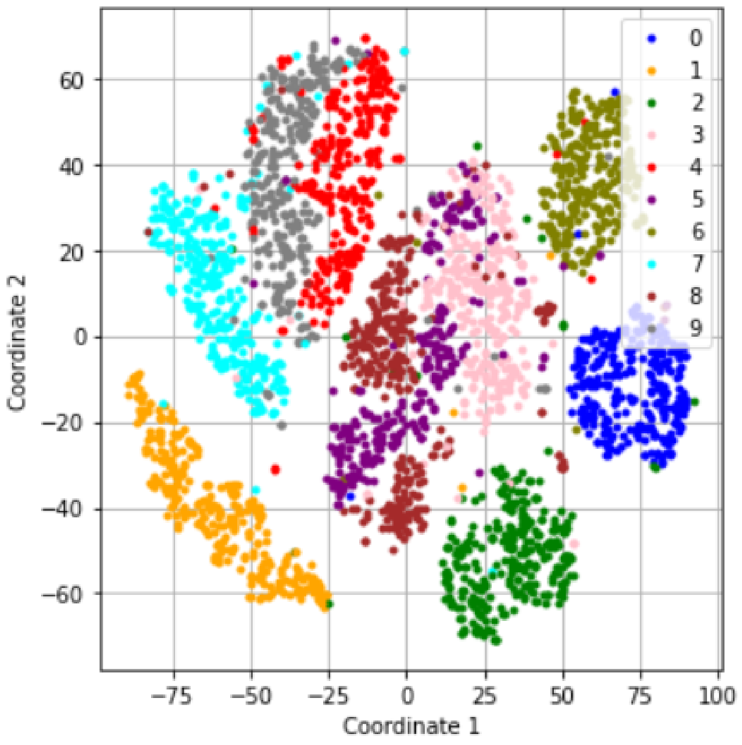


Figure 8: encoded space

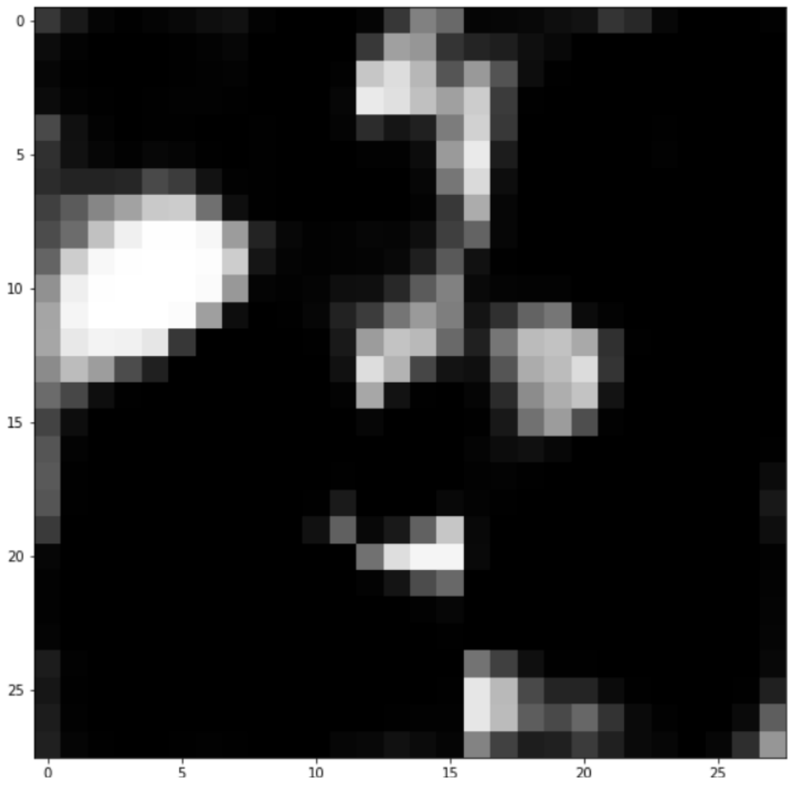


Figure 9: new sample tentative