# Homework 4

Anna Braghetto - 1205200

19/02/2020

## Introduction

The aim of the proposed homework is to build, train and test an autoencoder with PyTorch, given the MNIST digits dataset. In particular, the hyperparameters of the model are looked for by performenig a random search using the $k$-fold cross validation. Defined the best model, the autoencoder is trained and tested on both original and corrupted images. Then, to improve the learning, a denoising autencoder is also implemented. Finally, the generative capabilities of the trained autoencoder is analyzed and then compared with the results obtained through a variational autoencoder. Furthermore, since the needed computational power to perform the random search with many configurations is too high to use CPU, everything is perfomed with CUDA in Google Colaboratory.

## Dataset

The MNIST dataset contains 60000 images, representing handwritten digits, and their corresponding labels from 0 to 9. In particular, the downloaded dataset is already divided into training (50000) and test (10000). Furthermore, to perform the test on corrupted images and also to train the denoising autoencoder, two different transformations, that permit to introduce noise, are defined. The first one permits to add gaussian noise with an arbitrary mean and variance, while the second one permits to add occlusions (i.e. deleting a portion of the image). In Fig. 1 it is possible to observe the two kinds of corruptions applied on two sample images, near the orginal ones.

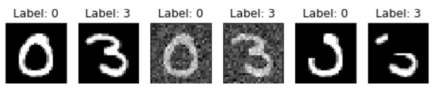

Figure 1: Original images (on the left), corrupted images with Guassian noisse with mean set to 0 and standard deviation set to 0.2 (in center) and corrupted images with occlusion (on the right).

## Model

Many hyperparameters are tested in order to define the best model. To explore many configurations, a random search with $k$-fold cross validation is performed.

In particular, the autoencoder is built by defining an encoder and decoder with an arbitrary dimension of the latent space, and the training is performed by using, as loss function, the *MSE loss*, that permits to minimize the reconstruction error, and the *Adam optimizer* that permits also to apply the $L_2$ *regularization*. Furthermore, to avoid overfitting, an early stopping is also implemented: if the validation loss increases for many consecutive epochs, the training stops.

In particular, the random search is performed fixing the latent dimension to 6 in order to obtain just the best learning rate and weight decay to train the model.

### Random Search

The considered learning rates and weight decays are the following.

- **Learning rate**: 19 different values of learning rate are considered $\in [10^{-4}, 2\cdot10^{-4}, 3\cdot10^{-4}, 4\cdot10^{-4}, 5\cdot10^{-4}, 6\cdot10^{-4}, 7\cdot10^{-4}, 8\cdot10^{-4}, 9\cdot10^{-4}, 10^{-3}, 2\cdot10^{-3}, 3\cdot10^{-3}, 4\cdot10^{-3}, 5\cdot10^{-3}, 6\cdot10^{-3}, 7\cdot10^{-3}, 8\cdot10^{-3}, 9\cdot10^{-3}, 10^{-2}]$
- **Weight decay**: 19 different values of weight decay are considered $\in [10^{-5}, 2\cdot10^{-5}, 3\cdot10^{-5}, 4\cdot10^{-5}, 5\cdot10^{-5}, 6\cdot10^{-5}, 7\cdot10^{-5}, 8\cdot10^{-5}, 9\cdot10^{-5}, 10^{-4}, 2\cdot10^{-4}, 3\cdot10^{-4}, 4\cdot10^{-4}, 5\cdot10^{-4}, 6\cdot10^{-4}, 7\cdot10^{-4}, 8\cdot10^{-4}, 9\cdot10^{-4}, 10^{-3}]$

A total of 35 combinations are defined by performing a random choice over the possible values of the two parameters.

## Cross Validation

The *k*-fold cross validation is perfomed on the training set considering the 35 configurations introduced above. The training set is divided into 4 folds: for each fold the autoencoder is trained in the union of the other 3, for 40 epochs, and validated in the selected one. The best configuration correposponds to the one with the lowest validation loss and its parameters are: learning rate equals to $4 \cdot 10^{-3}$ and weight decay equals to $2{\cdot}10^{-5}$.

## Dimension of Latent Space

To analyze how the size of the hidden layer affects the reconstruction perfomance, the autoencoder is trained by systematically increasing the dimension of the latent space. In particular, in order to avoid overfitting, the training, for all the considereded dimensions, is performed, with 80 epochs, by dividing the training set into two subsets: one training itself (80%) and one to validate the model (20%).

The results of the analysis are shown in Fig. 2, where it is possible to observe the test loss as function of the dimension of the latent space.

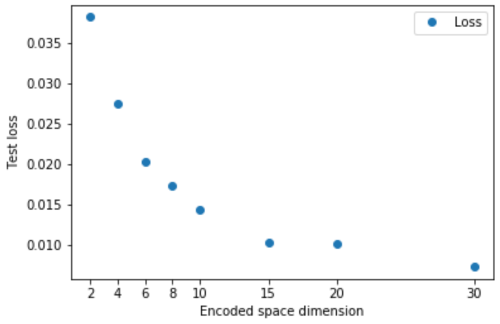

Figure 2: Test loss as function of the dimension of the latent space: as expected, the loss on the test set, decreases if the dimension increases. This is the expected behaviour because, by increasing the size of the hidden layer, the encoding occurs into an higher number of units, consequently, there is a lower loss of information, allowing to make a better reconstruction of the images.

## Final Training

Fixed the dimension of the latent space to 6, the final training is performed, as described above, using training and validation sets to avoid overfitting, with 80 epochs.

In Fig. 3, the training and validation loss, and the reconstruction of one sample, original and corrupted are shown. In particular the results obtained for the Gaussian noise refers to mean set to 0 and standard deviation set to 0.2.

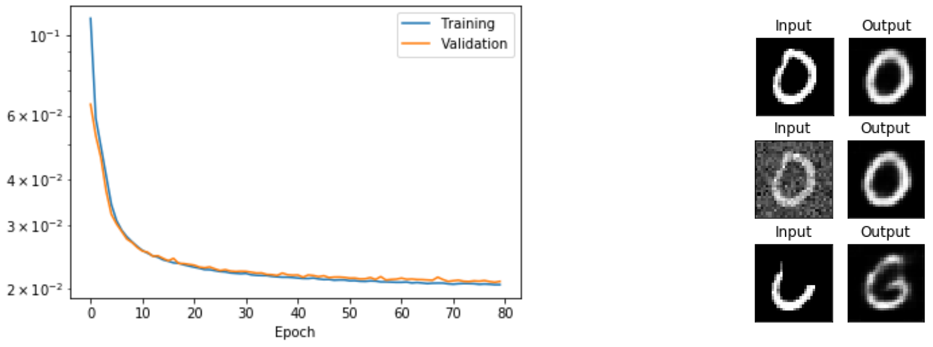

Figure 3: On the left, there are the training and validation loss that are really close, showing that there is not overfitting. On the right, there is the reconstruction for the same sample without corruptions and with corruptions for the autoencoder. The first belongs the original dataset and shows a good reconstruction, the second one belongs the corrupted dataset with guassian noise and shows a good reconstruction as well, while the third one belongs the corrupted data with occlusions and shows worse reconstruction, really different from the original one.

The loss on the original and corrupted test datasets are computed and collected in Tab. 1.

| Original Test | Guassian Test | Occlusion Test |
|---|---|---|
| 0.021 | 0.023 | 0.060 |

Table 1: Loss on test datasets for the autoencoder.

As shown in Fig. 3 and in Tab. 1, the autoencoder perform godd reconstructions in the original and corrupted dataset with guassian noise (the loss for different noise level is shown in Fig. 5), while in case of occlusions, it is really difficult to reconstruct the correct pattern.

## Denoising Autoencoder

In order to improve the learning, a denoising autoencoder is implemented. In particular the denoising autencoder learns to reconstruct corrupted images given the original ones. Thus, to perform the training it is necessary to build a new corrupted training and validation sets, with Gaussian noise (with mean set 0 and standard deviation set to 0.5) and occlusions: the kind of noise applied to each sample is chosen randomly between the provided two.

In Fig. 4, the training and validation loss and the reconstruction of one sample, original and corrupted are shown. In particular the results obtained for the Gaussian noise refers to mean set to 0 and standard deviation set to 0.2.

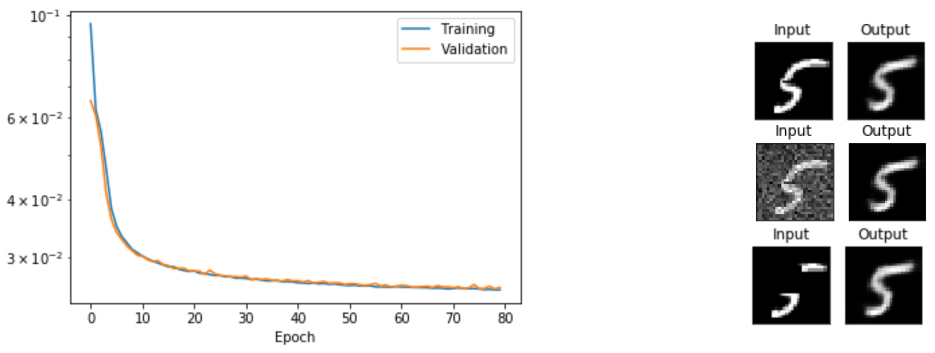

Figure 4: On the left, there are the training and validation loss that are really close, showing that there is not overfitting. On the right, there is the reconstruction for the same sample without corruptions and with corruptions for the autoencoder: in all the cases the reconstruction is good.

The loss on the orginal and corrupted test datasets are computed and collected in Tab. 2.

| Original Test | Guassian Test | Occlusion Test |
|---|---|---|
| 0.022 | 0.022 | 0.027 |

Table 2: CLoss on test datasets for the denoising autoencoder.

Comparing the results of the encoder (Fig. 3 and Tab. 1) with the results of the denoisinig autoencoder (Fig. 4 and Tab 2), it is possible to observe and improvement in the reconstruction of samples with noise. In particular, there is an high improvement in the case of the images with occlusions: the denoising autoencoder is able to reconstruct the pattern even if parts of the original image are removed.

In particular, many noise levels are tested for the Gaussian noise. The results for both the autoencoder and the denoising autencoder are shown in Fig. 5.

## Generative Capabilities

In order to generate good samples, a variational autoencoder is trained. This permits to obtained a more regular latent space in which it is possible to sample points through which generates new images.

### Variational Autoencoder

At first, the structure of the autoencoder is changed in order to take into account that, from the encoded input, it is obtained a distribution (*normal*) in the latent space, where a single point is sampled and then decoded, obtaining in this way the reconstructed image.

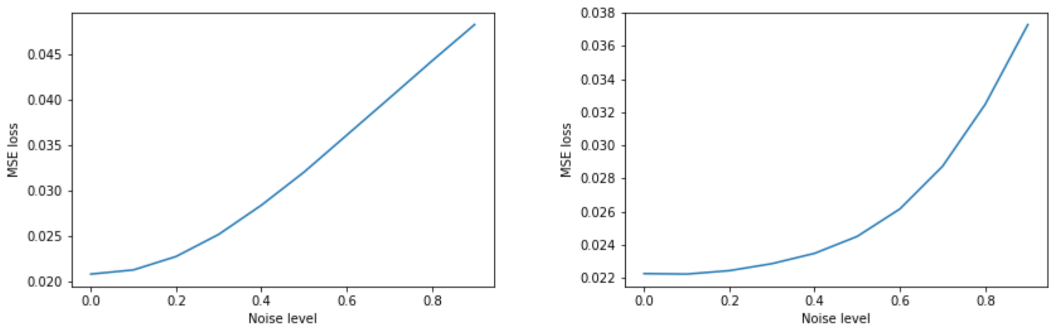

Figure 5: Test loss on the corrupted images with Gaussian noise at many levels for the autoencoder (on the left) and for the denoising autencoder (on the right). As expected, the loss increases with the increase of the noise level, showing a slower growth for the denoising autencoder.

Furthermore, to train the variational autoencoder, it is also necessary to the define the loss function to minimize. In particular it has two components: a reconstruction term and the Kullback-Leibler divergence between the latent distribution and the normal one. The first term is responsible for the good reconstruction of the image, while the second is responsible for the regularization of the latent space. In particular, in order to have a loss comparable with the other models, it is normalized over the batch size (512) and the dimension of image (784).

In order to simplify the learning, the training is performed just on the original dataset, without any corruptions, with the same parameters obtained with the random search. In Fig. 6 the training and validation loss and the reconstruction for one sample are shown.

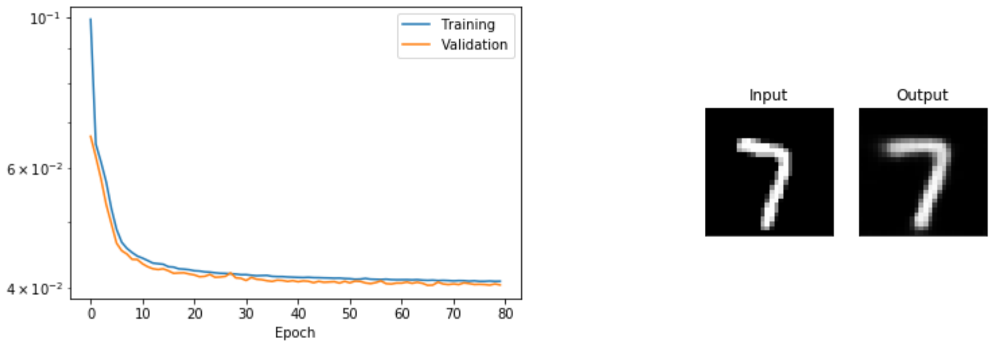

Figure 6: On the left, there are the training and validation loss that are really close, showing that there is not overfitting. On the right, there is the reconstruction for one sample: the reconstructed image is good.

Furthermore, the loss on the test dataset (without corruption) is computed:

| **Original Test** |
|---|
| 0.040 |

Table 3: Loss on test dataset for the variational autoencoder.

As observed in the results of the variational autoencoder (Fig. 6 and Tab. 3), the reconstruction is good. Obviously the loss is bigger with reference to the ones computed above, due the fact that it is also considered a regularization term.

## Samples Generation

Defined the models, the generative capabilities of the autoencoder and of the variational autoencoder are compared.

At first, the latent space is shown, in Fig. 7, for 2000 points in both cases: since it is hard to show for 6 dimensions, the t-SNE is applied to the points in order to show them in a bidemensional plane.

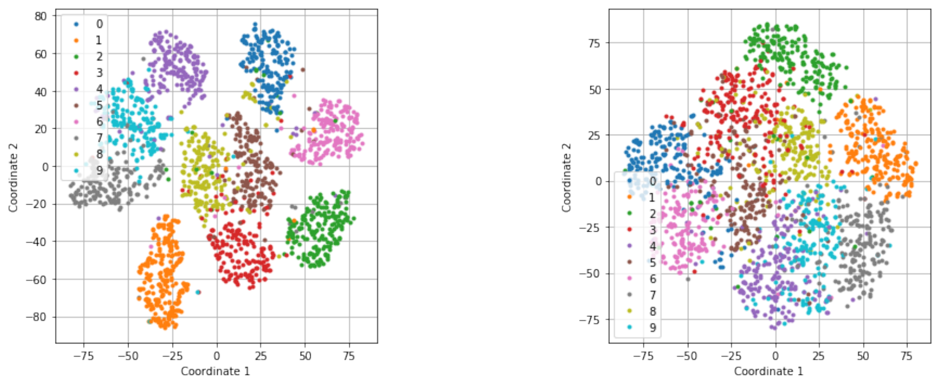

Figure 7: On the right, it is shown the latent space for the autoencoder that builds isolated clusters for each number, while, and on the left, it is shown the latent space for the variational autoencoder that builds a denser space with contiguous clusters.

Furthermore, random codes insides spaces are sampled and decoded to generate new samples. For both models, 100 samples are generated and shown in Fig. 8.

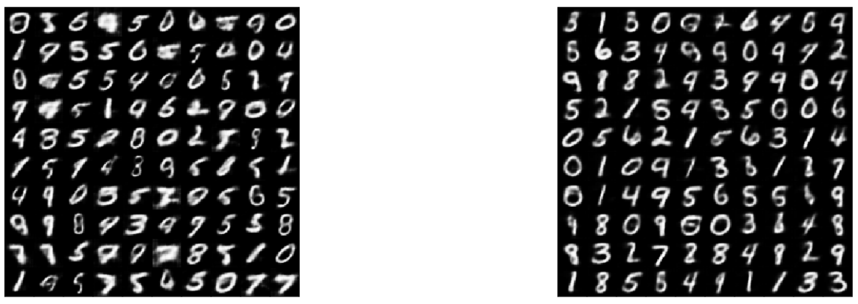

Figure 8: 100 random new samples for the autoencoder (on the left) and for the variational autoencoder (on the right). It is possible to observe that for the autoencoder, some samples are meaningless, reflecting the presence of points in the latent space that the decoder does not know how to reconstruct well. While, for the variational autoencoder, all the new samples are good.

Finally, in order to observe how the reconstruction changes, moving from point to another, the steps between two points in the latent space is shown for both models, in Fig. 9.

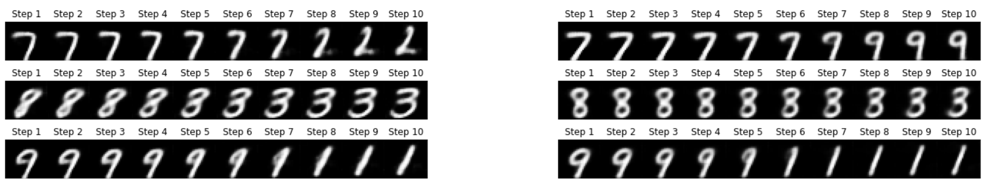

Figure 9: Transitions between two points for the autoencoder (on the left) and for the variational autoencoder (on the right). The results are similar.

Through both the models it is possible to generate good samples, but as expected, there is an improvement by using the variational autoencoder.