Homework 4

Francesco Grimaldi, 1206035
Department of Mathematics, University of Padua

Neural Networks and Deep Learning
Professor: Dr. Alberto Testolin
Assistant: Dr. Federico Chiariotti

November 2019

1 Introduction

This report describes the work done for the 4th home assignment of the course in Neural Networks and Deep Learning.

In this homework it has been build a Variational Autoencoder (VAE) that try to perform dimensionality reduction of the MNIST digit dataset. Various VAE have been created in order to explore the best dimension of the latent space and to test how robust the model is when dealing with noisy data (white noise).

2 Dataset

Initially, the dataset has been augmented by adding the same images but with random gaussian noise. This procedure made the model to be more robust and to learn a more consistent latent space. The augmented data has been generated as follow:

$$A' = A + \lambda N \tag{1}$$

where:

- *A'*, *A* are respectively the augmented and the initial (a digit image) inputs;
- λ is a scalar values taking value 0.1 and 0.3;
- N is a matrix (28x28) having elements sampled from a Normal Distribution.

After this procedure, the dataset has been divided in training and test with proportion 0.8/0.2. Those two dataset have been divided in batches of size equal to 128.

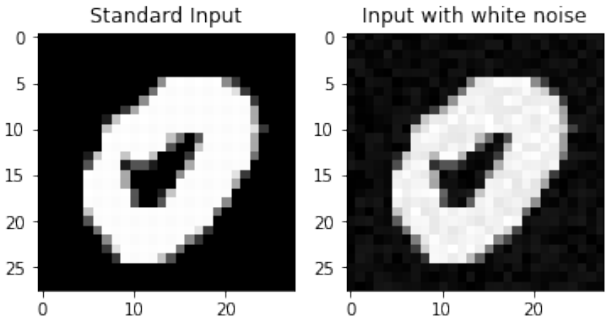

Fig 1. Initial Data and Augmented Data (λ = 0.1)

3 The model

In order to create a VAE some modification have been done to the code of the Autoencoder class and of the training phase, that was given in the lab.

3.1 VAE architecture

First, the hidden layer of the latent space have been divided in two hidden layers learning the mean μ_x and the variance σ_x of a Normal Distribution N w.r.t an input x, such that the latent space is represented by a distribution.

Second, the input of the decoder is now sampled from the distribution with the parameters given by the above cited two layers.

3.2 Loss Function

The loss function has been changed in the sum of two functions: the reconstruction loss (computed with binary cross entropy rather than the MSE) and the regularization term (expressed with the Kullback-Leibler divergence).

$$\mathcal{L}(x; w) = ||x - x'||^2 + KL[N(\mu_x, \sigma_x), N(0, 1)] \tag{2}$$

with:

- x, x' being respectively the input and the ouput of the model;
- $KL[\cdot, \cdot]$ being the Kullback-Leibler divergence function;
- μ_x, σ_x being respectively the mean and the variance of the latent distribution given by our input x.

4 Parameters Search

4.1 CNN vs Feed-Forward

The first tests were done using either a VAE with Convolutional Layers (as given in the lab) or simple VAE with normal feed-forward architecture (1 hidden layer in the encoder, 2 hidden layer for the latent space and 1 hidden layer for the decoder).

It has been seen, weirdly enough, that the VAE with standard feed-forward layers was working faster and better than the CNN. For this reason it has been decided to go ahead with the standard Feed-Forward architecture

4.2 Learning Rate and Weight Decay

A fast grid search has been performed to check for the best initial hyper parameters for the optimization algorithm *Adam*. Consistent hyper parameters were found to be a *learning rate* of 1e-3 and a *weight decay* of 1e-5.

4.3 Number of hidden neurons

For the number of hidden neurons of our latent space different values have been tried, ranging from 2 to 80.
The validation loss decreased as the number of neuron increased, until the threshold of 20/40 hidden neurons.

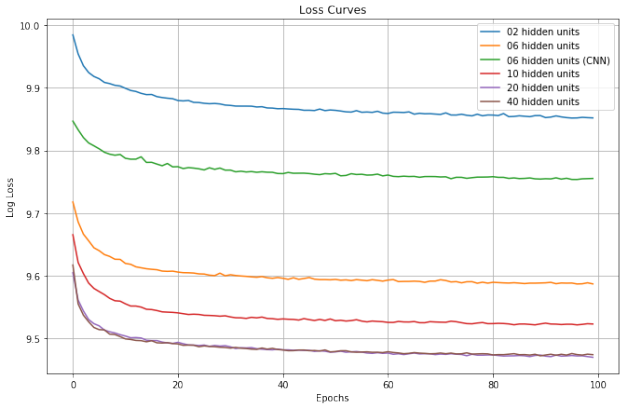

Fig 2. Validation Loss of VAE with different number of neurons on the latent space. It can be seen that a CNN-VAE performs worse that a FNN-VAE, while the smallest loss was reached at 20/40 hidden neurons.

5 Model Results

The model, even if trained with a Binary Cross Entropy plus Kullback-Leibler divergence function, was tested with a MSE loss function.

It achieve a MSE of 0.012 on the original data and a MSE of 0.013 when tested with validation data with additional white noise.

5.1 Reconstruction Error with noise

The model manages also to reconstruct images with discrete discrete level of noise (see *fig. 3*) even if was not trained with such noisy data, in fact the model manage to reconstruct images very well even with a medium/high level of noise ($\lambda = 1$, see eq. 1) even if trained with images with much less noise ($\lambda = 0.3$).

Moreover, the model seems to be able to deal with badly written digits by reconstructing them in a better way, taking out extra pixel (see second row of *fig. 3*) or filling some gaps (see first row of *fig. 3*).

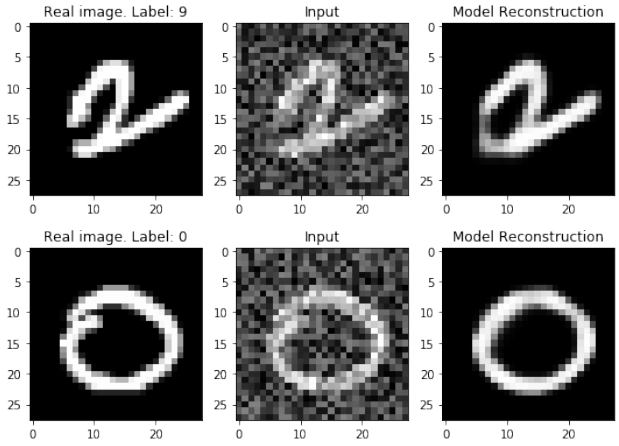

Fig 3. Examples of images reconstruction. In the left we can see the real image, in the middle there is the noisy input and to the right the model reconstruction

5.2 Extreme noise condition

The model have also be tested with higher level of noise ($\lambda = 2$), managing to reconstruct the true input.
The reconstruction was pretty consistent (see *fig. 4*) but there were cases were the reconstruction led to different digits (e.g. a very noisy 0 could be reconstructed as an 8 or viceversa).

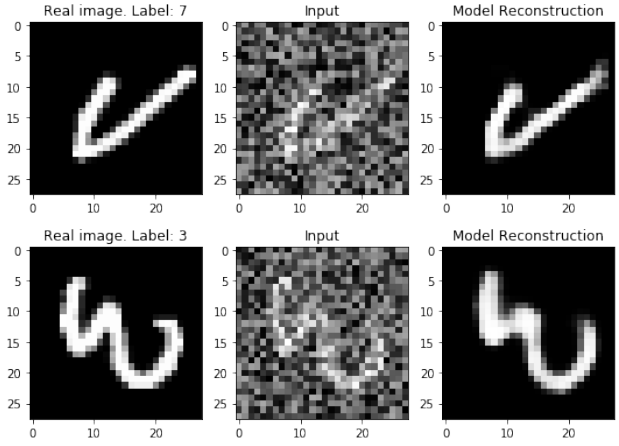

Fig 4. Reconstruction with extreme noise ($\lambda = 2$)

5.3 Flipping pixels and occlusion

The model seemed not to be so much robust when dealing with other forms of noises like flipping or occlusion.
In fact, when the input is not corrupted with white noise but with some pixel flipped from 0 to 1 or vice-versa (see *fig. 5*) the MSE is much higher reaching value of 0.0403.

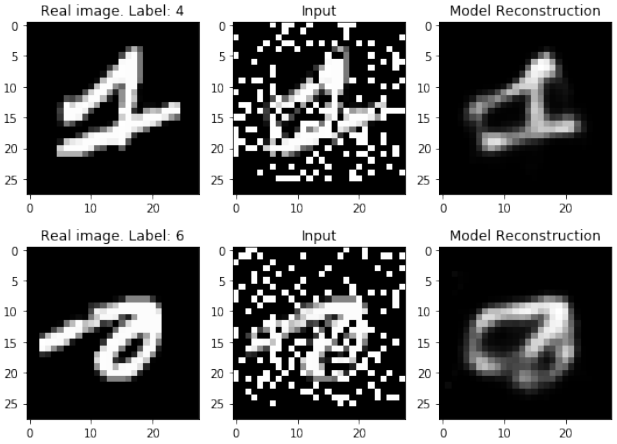

Fig 5. Reconstruction when some pixels are flipped. In the first row we can see a successful reconstruction od the digit 1, meanwhile in the second row the reconstruction goes badly, in this case the model try to reconstruct something that can be seen as an 8 instead of a 9

The same is when occlusion is performed on part of the images (see *fig. 6*), also in this case the model isn't performing perfectly. Here the MSE is about 0.0435.

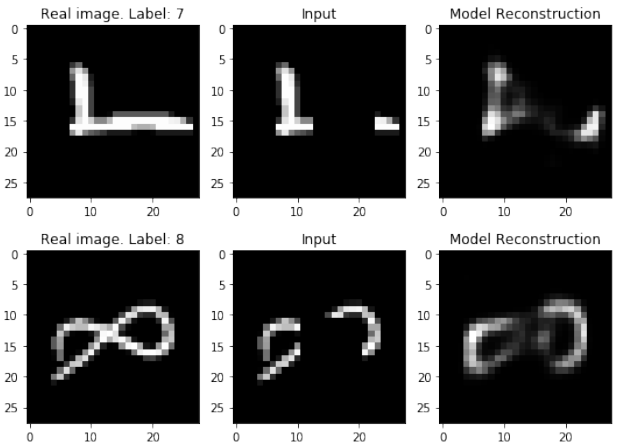

Fig 6. Reconstruction when a part of an image is hidden. In the first row we can see a 7 that is tried to be reconstructed as something similar to a 3, while in the second we can see a quite good reconstruction of the digit 8.

6 Sampling from codes in the latent space

It has been tried also to sample directly from the latent space too see if the learned latent space was regular enough. It has been seen that sampling from the same normal distribution given by fixed μ_x and σ_x produced very consisted outputs. Meanwhile sampling by different distributions (by changing μ and λ just a little) produced a consistent shift of the decoded output (see *fig 7.*).

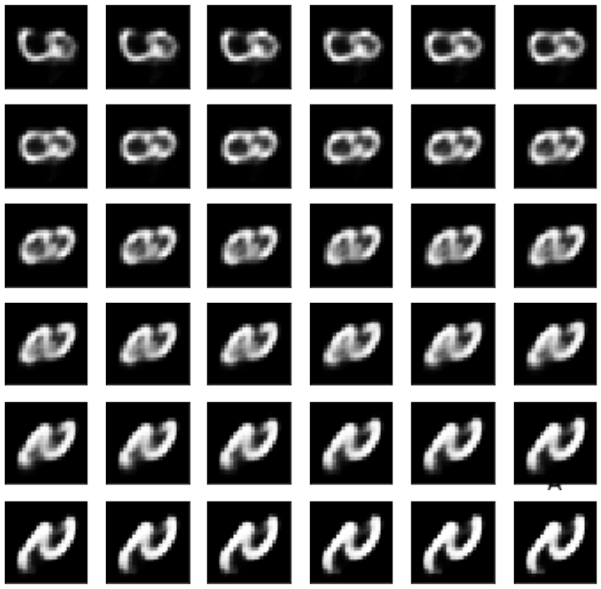

Fig 7. Reconstruction by sampling from the latent space. As the sampled latent vector changes in one direction so does the reconstructed output. We can see that starting from a 9 we slowly shift to an 8 and to what seems a 2.

7 Guide to the code

The zip file contain 5 files:

1. *VAE.py* is the script to build the model;
2. *Dataset.py* is the script to pre process the data;
3. *model.json* is a file containing the params of the model;
4. *trained_model.py* is the script to run;
5. this pdf file.

The *trained_model.py* should work by writing *python trained_model.py "MNIST.mat"*, where *MNIST.mat* is a dictionary with the images under the label *input_images*.

The script load the images, pass them to the model and compute the *MSE loss* between the original and the reconstructed image.

It will also generate a .mat file called *recon_MNIST.mat*, a dictionary like the previous file, where every output is saved under the dictionary label *input_images*.