**Ivancich Stefano 1227846**

**Neural Network and deep learning course 2020/21**

**Homework 2**

1. **Introduction**

In this homework we have to implement and test neural network models for solving unsupervised problems. The homework is based on images of handwritten digits (MNIST).

The **basic tasks** for the homework require to test and analyze the **convolutional autoencoder** implemented during the Lab practice, explore the use of advanced optimizers and regularization methods. Learning hyperparameters should be tuned using appropriate search procedures, and final accuracy should be evaluated using a cross-validation setup.

Reporting the **trend of reconstruction loss** and some examples of image reconstruction.

Implement the **denoising autoencoder**.

Use the convolutional encoder as base model for the **classification task** and compare the classification accuracy with Homework 1.

Explore the **latent space structure** (e.g., PCA, t-SNE) and generate new samples from latent codes.

Implement **variational (convolutional) autoencoder**.

1. **Convolutional Autoencoder**

For the encoder architecture is composed by 3 convolutional layers followed by 2 linear layers all with Relu and Dropout. The decoder architecture is symmetric.

We didn’t use **cross validation** in this case, since we have 60.000 images, we did not think that we can overfit in a such simple task.

A **grid search** of the following hyperparameters was ran:

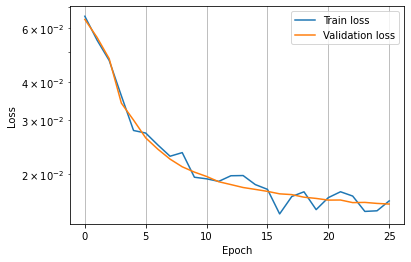
* **encoded\_space\_dim:** 2,4,10,128
* **Conv1:** 8,16,32, 64 filters of size 3x3
* **Conv2:** 8,16,32, 64 filters of size 3x3
* **Conv3:** 8,16,32, 64 filters of size 3x3
* **FC1 number of neurons:** 32,64,128
* **FC2 number of neurons:** =**encoded\_space\_dim**
* **Layers activation:** ReLu for all except the last layer of the encoder that has no activation.
* **Dropout:** 0, 0.25, 0.5, 0.75
* **Optimizer:** Adam
* **Learning rate:** 0.1, 0.01, 0.02, 0.001
* **Regularization:** "L2" with values 1e-3, 1e-4, 1e-5 and 0 (no regularization)
* **Max epochs:** 3000 (we did not choose to tune this value, because the early stopping will take care of it)
* **Early stopping:** max 10 epochs without improvement

The best hyperparameters for the turned out to be:

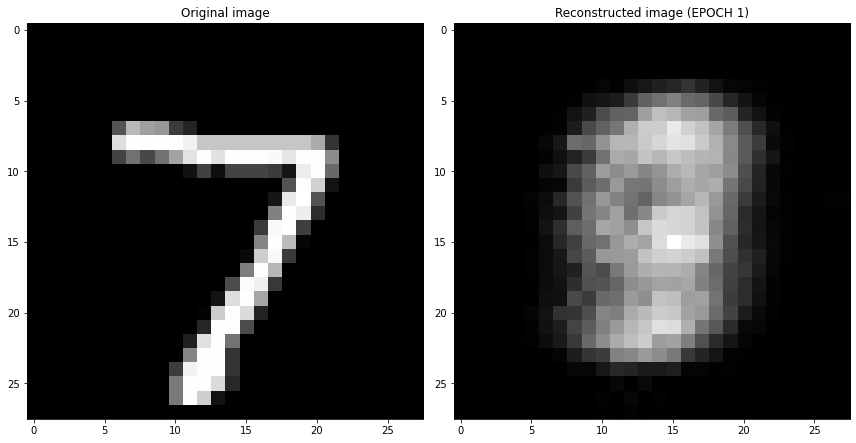
* **encoded\_space\_dim: 10**
* **Conv1:** 8 filters of size 3x3
* **Conv2:** 16 filters of size 3x3
* **Conv3:** 32 filters of size 3x3
* **FC1 number of neurons:** 64
* **Learning rate:** 0.001
* **Regularization:** 1e-5 (L2)
* **Dropout:** 0. Strangely the dropout was slowing a lot the learning and also having bad loss and bad reconstruction.

**Reconstruction loss Results:**

* Train Loss: 0.016
* Val Loss: 0.016



Here some ……………



**Result**

present the simulation results.

* 1 pt: implement and test (convolutional) autoencoder, reporting the trend of reconstruction loss and some examples of image reconstruction
* 1 pt: explore advanced optimizers and regularization methods
* 1 pt: optimize hyperparameters using grid/random search and cross-validation
* 1 pt: implement and test denoising (convolutional) autoencoder
* 1 pt: fine-tune the (convolutional) autoencoder using a supervised classification task (you can compare classification accuracy and learning speed with results achieved in homework 1)
* 1 pt: explore the latent space structure (e.g., PCA, t-SNE) and generate new samples from latent codes
* 2 pt: implement variational (convolutional) autoencoder or GAN

Encoder(

(encoder\_cnn): Sequential(

(0): Conv2d(1, 8, kernel\_size=(3, 3), stride=(2, 2), padding=(1, 1))

(1): ReLU(inplace=True)

(2): Dropout2d(p=0, inplace=False)

(3): Conv2d(8, 16, kernel\_size=(3, 3), stride=(2, 2), padding=(1, 1))

(4): ReLU(inplace=True)

(5): Dropout2d(p=0, inplace=False)

(6): Conv2d(16, 32, kernel\_size=(3, 3), stride=(2, 2))

(7): ReLU(inplace=True)

(8): Dropout2d(p=0, inplace=False)

)

(flatten): Flatten(start\_dim=1, end\_dim=-1)

(encoder\_lin): Sequential(

(0): Linear(in\_features=288, out\_features=64, bias=True)

(1): ReLU(inplace=True)

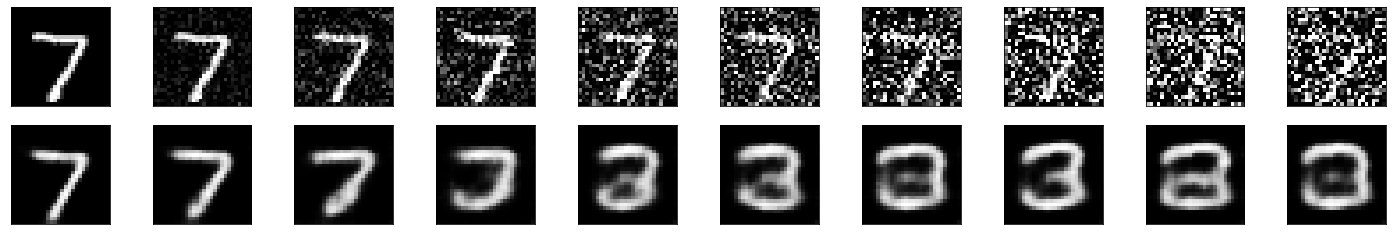
(2): Dropout(p=0, inplace=False)

(3): Linear(in\_features=64, out\_features=10, bias=True)

)

)

Normal autoencoder reconstruction:



Denoising Autoencoder reconstruction:



VAE:



**Classification**

We added a fully connected layer composed of 10 neurons (one per class 0..9) on the encoder, then we freezed all layers except the last 2.

We obtained very good result without any fine tuning at the first training (96.9% test accuracy against the 98.5% of the previous homework). That means that unsupervised pre training is a powerful tool.

After some brief fine tuning we achieve the best results with learning rate =…..

Train Loss: 0.109

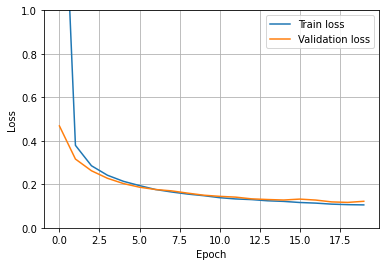
Val Loss: 0.121

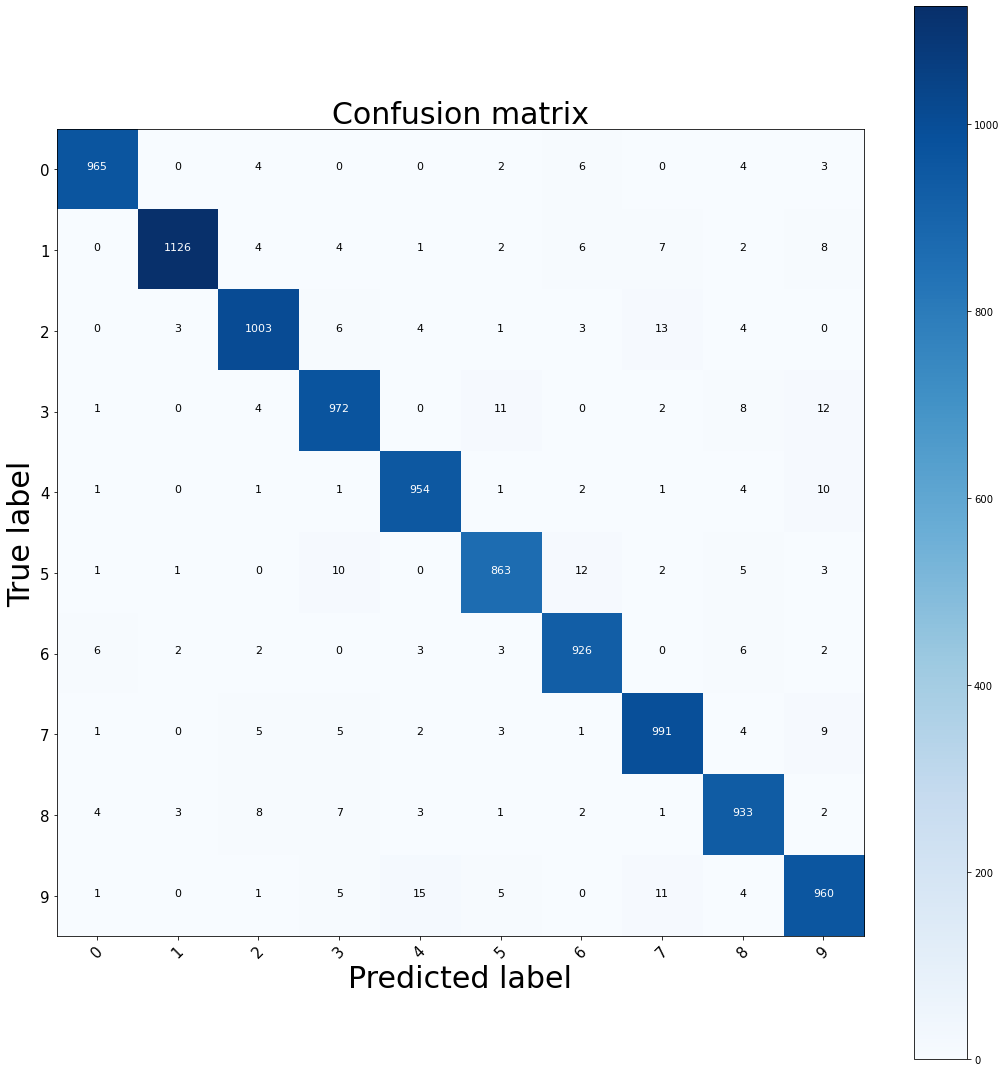
Test Loss: 0.099

Train Accuracy: 0.966

Val Accuracy: 0.966

Test Accuracy: 0.969





Possible questions:

* What is an autoencoder?