UNIVERSITY OF PADOVA

Homework 2 Report

Subject: INP9086459 - NEURAL NETWORKS AND DEEP LEARNING 21/22
Professor: ALBERTO TESTOLIN
Student: Menglu Tao Nº: 2041389

1. Introduction

In this homework we need to learn how to implement, test and analyze the **convolutional autoencoder** implemented during the Lab practice. We should 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,fine-tune the (convolutional) autoencoder using a **supervised classification task**, and compare classification accuracy and learning speed with results achieved in Homework 1.
explore the **latent space structure** (e.g., PCA, t-SNE) and generate new samples from latent codes,more advanced tasks will require the e**xploration of denoising** and **variational** / adversarial architectures.

2. Convolutional autoencoder

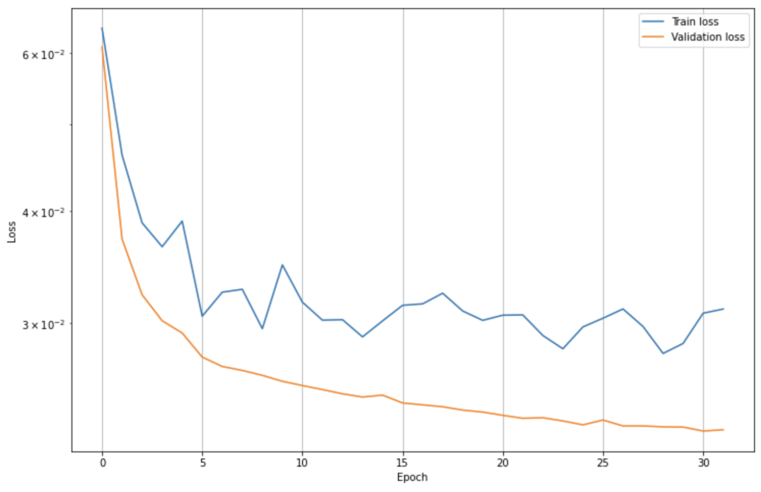
A convolutional autoencoder is composed by an encoder and decoder. The encoder is composed by 3 convolutional layers and 2 linear layers, decoder has similar structure, but starts with 2 linear layers then followed by 3 convolutional layers.
Configurations of the encoder (using grid search):

- **Encoded_space_dim**: 10
- **First convolutional layer**: 8 or 16
- **Second convolutional layer**: 16 or 32
- **Third convolutional layer**: 16 or 32
- **First FC layer** :32
- **Second FC layer**: equals to encoded_space_Dim
- **Activation method**: Relu
- **Optimizer**: Adam
- **Learning rate**: 1e-3
- **Regularization**: 1e-5
- **drop out**:0.2
- **Criterion**:MSELoss
- **Max epoch**: 50 epochs

- **Early stopping**: 8 epochs

And the results given to use the best combination is:

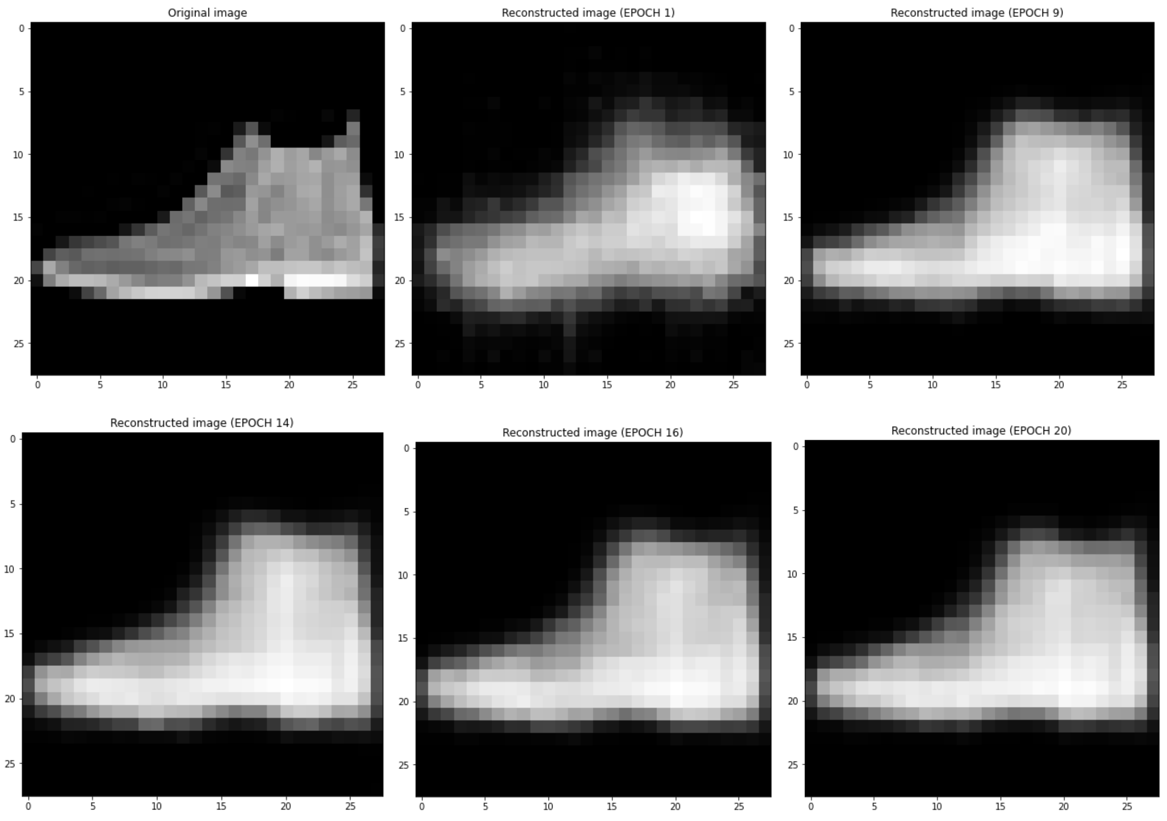
- **conv1**: 16
- **conv2**: 32
- **conv3**: 32
- **fc**: 32
- **lr**: 1e-3
- **l2**: 1e-5



And using these new parameters set, the new results are:

- **Train Loss**: 0.024
- **Val Loss**: 0.021
- **Test Loss**: 0.021

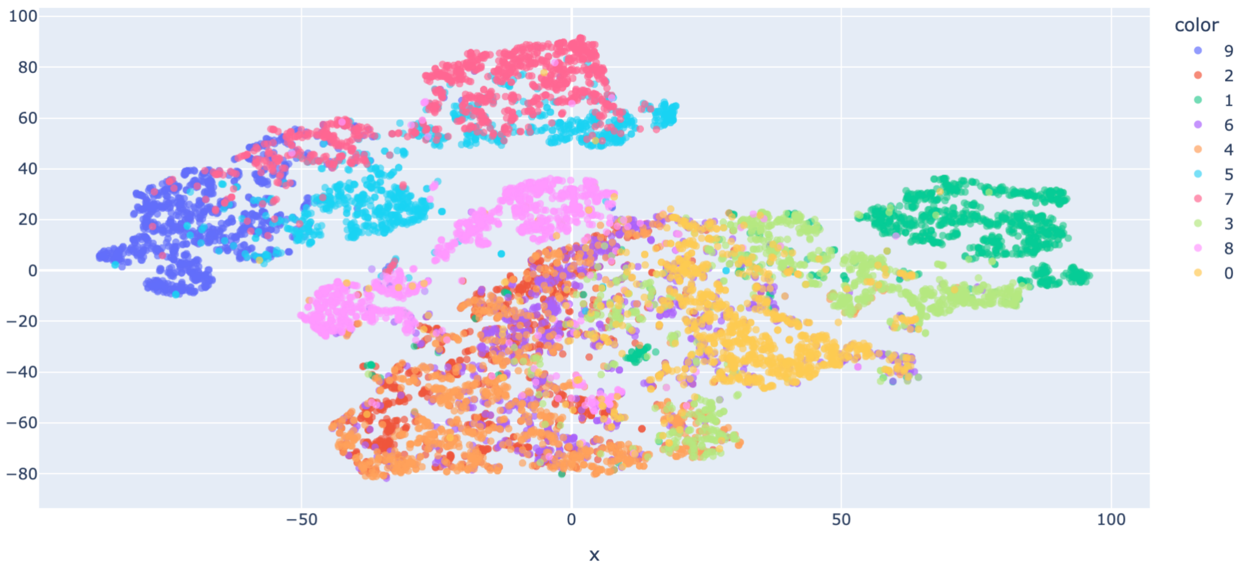
Reconstructed Images:


Original image
Reconstructed image (EPOCH 1)
Reconstructed image (EPOCH 9)
Reconstructed image (EPOCH 14)
Reconstructed image (EPOCH 16)
Reconstructed image (EPOCH 20)

3.Analysis

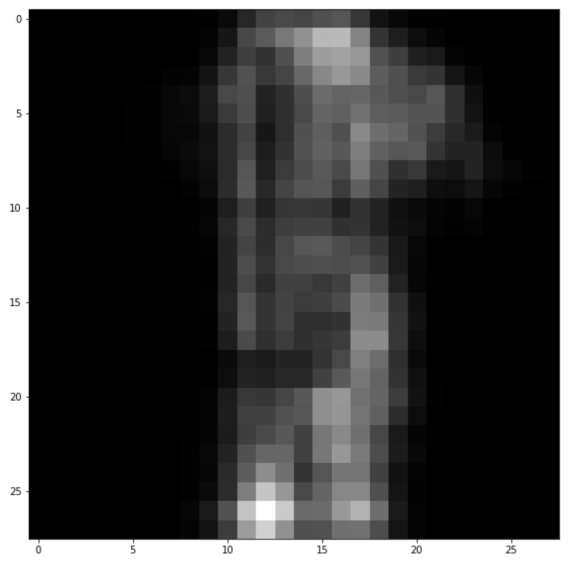
Latent Space Visualization

Here we choose to use t-SNE to show the cluster classification since its a 10 dimensional space, and we can clearly see some parts of clustering are good like the green parts(1,3) and pink(8), purple parts(9), while some are not that good, the orange one(4).



Generation new samples:

From this image, we can see it's not a very good one because it's hard to distinguish the type.

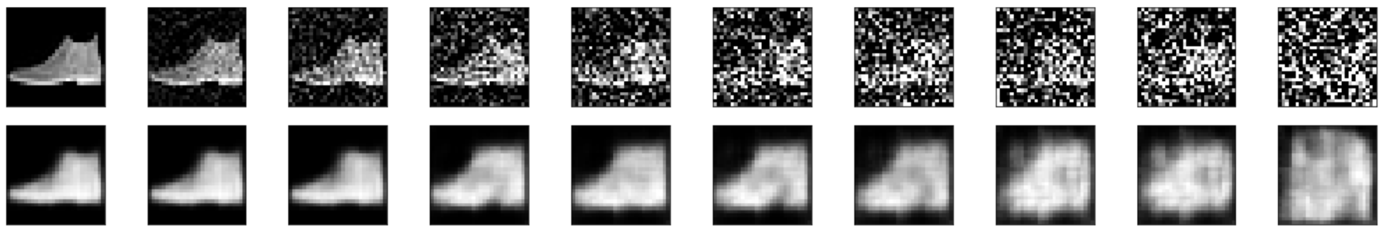


4.Convolutional denoising autoencoder

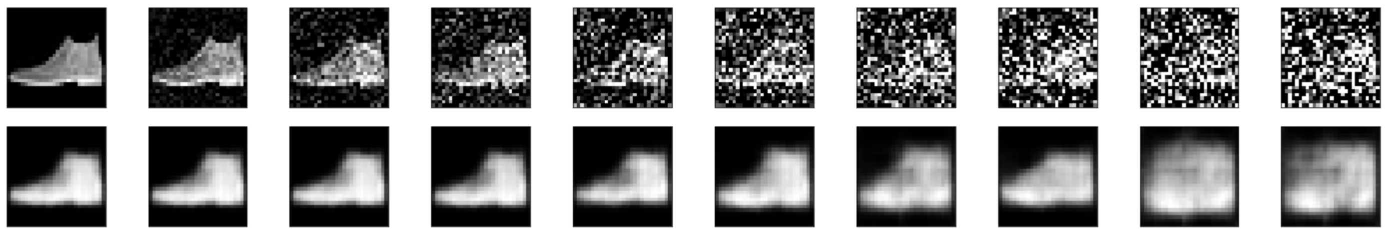
Here we're applying images with some gaussian noise (noisy_image = gaussian_noise(imgc, 0.1 * i)) and use them to train the denosing autoencoder model, and see if it has better result than the nomal one.

The parameters are the same as convolutional autoencoder.

Before training with noisy images to reconstruct images :

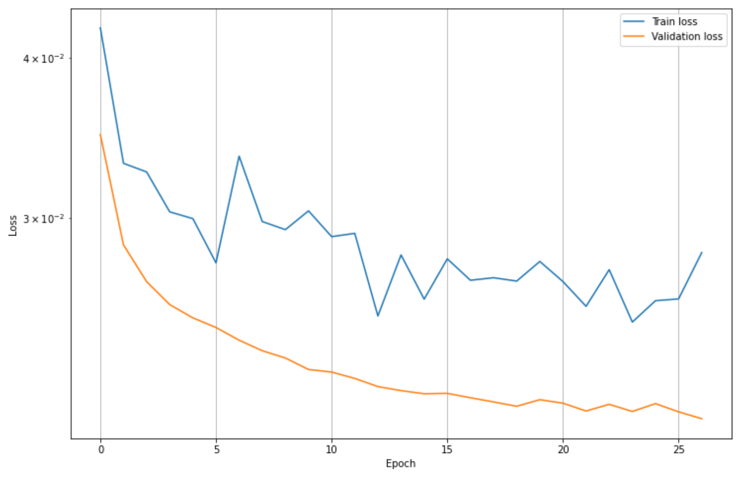


After training with noisy images to reconstruct images :



We can clearly see that the latter one has done a better job in reconstructing images with noise.

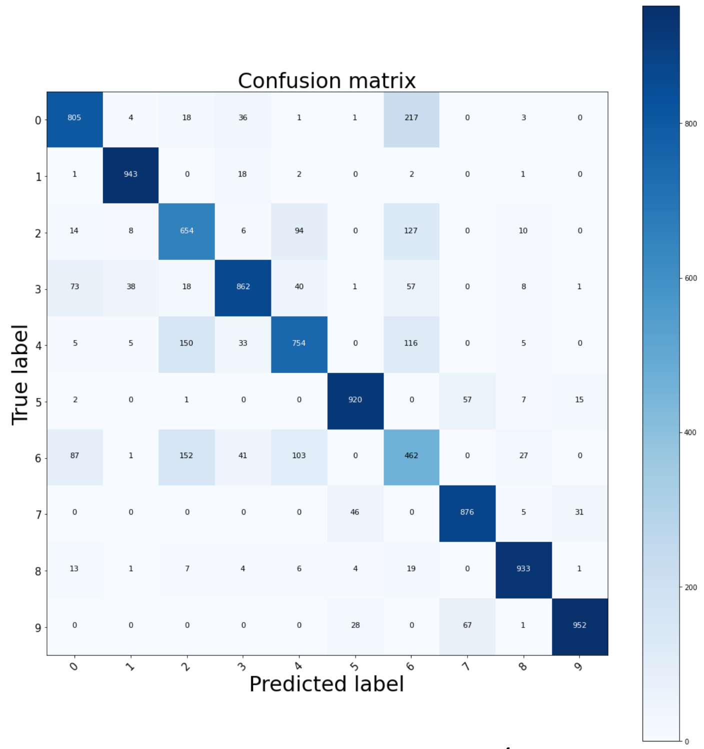
5.Supervised classification



Split the train datasetand use 80% for training, 20% for validation,and using the best parameters set below is the results"

- **Train Accuracy**: 0.805
- **Val Accuracy**: 0.827
- **Test Accuracy**: 0.816

since validation loss is lower than train loss, and train loss is still high when it finished, we could increase the number of epochs and probably obtain better results.
When this situation happens we can see that the model is not overfitted to the training data, which is good. The usage of dropout helped on this behavior. But because of the time to tune and run everything was high, so we didnt put more epochs.
Execution time: **10 mins**



	0	1	2	3	4	5	6	7	8	9
0	805	4	18	36	1	1	217	0	3	0
1	1	943	0	18	2	0	2	0	1	0
2	14	8	654	6	94	0	127	0	10	0
3	73	38	18	862	40	1	57	0	8	1
4	5	5	150	33	754	0	116	0	5	0
5	2	0	1	0	0	920	0	57	7	15
6	87	1	152	41	103	0	462	0	27	0
7	0	0	0	0	0	46	0	876	5	31
8	13	1	7	4	6	4	19	0	933	1
9	0	0	0	0	0	28	0	67	1	952

The **accuracy** achieved by HW1 is around **0.898** ,**execution time** is around **13 mins**, so we can say in this case regarding to accuracy using CNN is a better a solution than autoencoder with a little sacrifice of time consumption.

Seen from the confusion matrix, label 6 and label 2 are the most misclassificated cases.

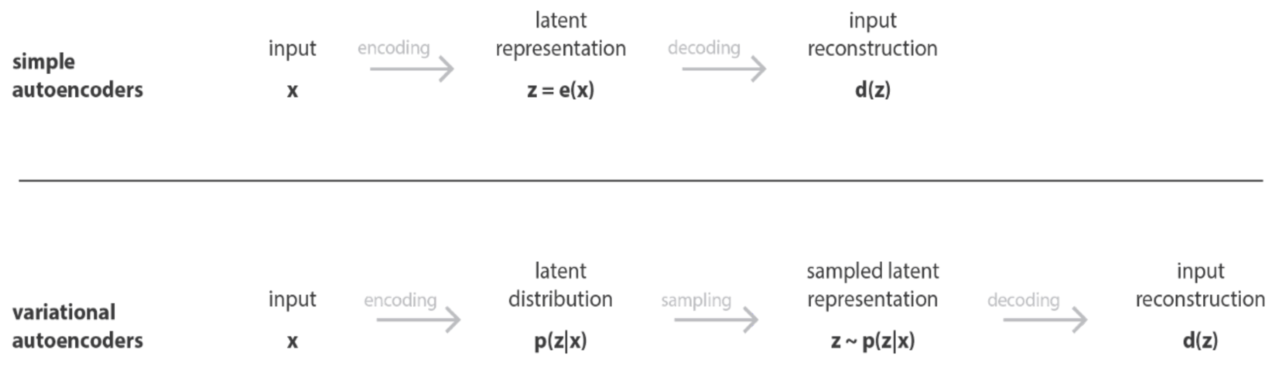
6.Variational autoencoder

The variational autoencoder is composed by 3 fully connected layer and Relu activation method. Decoder's structure is symmtic.

Parameters :

- **Encoder input**:784 features
- **Encoder FC1 neurons** : 512
- **Encoder FC2 neurons**: 256
- **Encoder Mean coding FC number of neurons**: 2
- **Encoder Variance coding FC number of neurons**: 2
- **Decoder input**: 2 features
- **Decoder FC1 neurons** : 256
- **Decoder FC2 neurons** : 512
- **Decoder output**: 784 features

Since in variational autoencoder the input is encoded as distribution over the latent space, then a point from the latent space is sampled from that distribution, and the sampled point is decoded and the reconstruction error can be computed, so the loss function is different with the normal encoder one.



In variational autoencoders, the loss function is composed of a reconstruction loss (that makes the encoding-decoding scheme efficient) and a regularisation term (that makes the latent spaceregular).

- Reconstruction loss: This loss compares the model output with the model input. We used **binary cross entropy**.
- Latent loss: This loss compares the latent vector with a zero mean, unit variance Gaussian distribution. The loss we use here will be the **KL divergence los**s.

$$Loss = L\left(x, \hat{x}\right) + \sum_{j} KL\left(q_j\left(z|x\right)||p\left(z\right)\right)$$

Below are some generated images: