Application of Unsupervised Deep Learning techniques in the MNIST digit Dataset

Delfina Iriarte (1231682)

[1]University of Padova, Physics of Data

Abstract. *In this work, the generation of new handwritten Digit* MNIST *Dataset is done by means of a Convolutional autoencoder. Original images are also obtained by applying a denoiser in the samples with random noise. Moreover, supervised techniques such as classification are study by applying transfer learning. Finally, a more advance architecture is investigated called Variational Autoencoder.*

1. Introduction

In this work, unsupervised feature learning is study by means of a convolutional autoencoders. Autoencoders (AEs) are methods that encode some input into a representation with low dimensionality, and then, via a decoding module, reconstruct this compact representation to match as best as possible the original input. The space formed by the transformation of the input into the code is often called latent space.

In particular, a convolutional autoencoder is applied and optimize in the MNIST Dataset using PYTORCH which is an exhaustive dataset containing handwritten digits from 0 to 9 using 28 × 28 images. Furthermore, imaging denoising is performed, as well as transfer learning. Finally, a more advance architecture is investigated called Variational Autoencoder.

2. Method

2.1. Convolutional autoencoder

A convolutional autoencoder basically consist of convolutional layers in the encoder and transposed convolutional layers in the decoder. In particular, Figure 5 presents the architecture of each of the component with the best parameters. As it can be seen, the encoder is composed by three convolutional layers and two linear layers and the decoder is simply the opposite. The activation function used in this case was the Rectified linear unit (ReLU) since it avoid vanishing problems in the gradient. Following standard autoencoders, the loss function chosen is the mean square error.

Due to the fact that the MNIST dataset is quite exhaustive, a cross validation technique is not implemented. However, in order to improve the performance of our model, a regularization technique is applied. Specifically, a Ridge regression regularization (L_2) is used by using the `weight_decay` variable. Furthermore, EARLY STOPPING and hyperparameter optimization was also used. In particular, the following parameters were optimize:

1. channel of the first convolutional layer `conv1`: $[8, 16, 32, 64]$
2. channel of the Second convolutional layer `conv2`: $[16, 32, 64]$
3. channel of the third convolutional layer `conv3`: $[32, 64]$
4. number of neurons of the first linear layer `fc`: $[32, 64]$
5. learning rate `lr`: $[1e-5, 1e-4, 1e-3]$
6. Regularization penalty `l2`: $[1e-5, 1e-4, 1e-3]$
7. Dropout rate `drop` :$[0.1, 0.5]$

Finally, it should be worth mentioning that the encoding space dimension was set to two so that much of the information is compress, however it could be also consider as an hyperparameter to optimize.

2.1.1. Latent Space Analysis

As it is known, the latent space is simply a representation of compressed data in which similar data points are closer together in space. Additionally, two dimensionality reduction tools are applied: Principal Component Analysis (PCA) and t-distributed stochastic neighbor embedding (t-SNE).

2.2. Fine tuning

The basic premise of transfer learning is simple: take a model trained on a large dataset and transfer its knowledge to a smaller dataset. The idea is the convolutional layers extract general, low-level features that are applicable across images — such as edges, patterns, gradients — and the later layers identify specific features within an image such as eyes or wheels.Since we have an already trained encoder, transfer learning techique can be applied which allows to get better performance of a classification task. The parameters of the encoder were ”freeze”, since no training were done for them, but the last two layers are not. Furthermore, an additional layer with 10 neurons, in order to classified the 10 digits, is added.

2.3. Denoising autoencoder

The model architecture of the autoencoder and the hyperparameters search are the same as for the one in Section 2.1. In this occasion, the input image correspond to a nosy image of the MNIST dataset that was obtained by adding some gaussian noise in which a new variable called `noise_amount` was added in order to control the amount of white noise in the data.

2.4. Variational Autoencoder

The classical architecture of the variational autoencoder is shown in Figure 1. As it can be observed, the main difference between the convolutional autoencoder is the latent representation where the mapping is probabilistic. In particular, we implemented a multivariate Normal distribution for the conditional distribution of the latent vectors given and input image ($q_\phi(z|x_i)$) and a multivariate Bernoulli distribution for the conditional distribution of images given the latent vector ($p_\theta(x|z)$). The encoder is simply given by two convolutionals layers whereas the decoder two transpose convolutionals layers.In particular, `mu` and `logvar` are standard fully connected layers that will represent mean and log-variance, respectfully. Using outputs of these layers will be then used to sample latent variable z. Now, since the sampling is a discrete process, in order to allow backpropagation, **parameterization trick** is used. As activation the sigmoid function is used as in the original paper. However, instead of SGD, Adam optimizer was used. The loss function consists of two terms. Reconstruction loss, for which was used `binary_cross_entropy`, and regularization loss or Kullback–Leibler divergence that will force z to be a normal distribution with mean=0 and std=1.

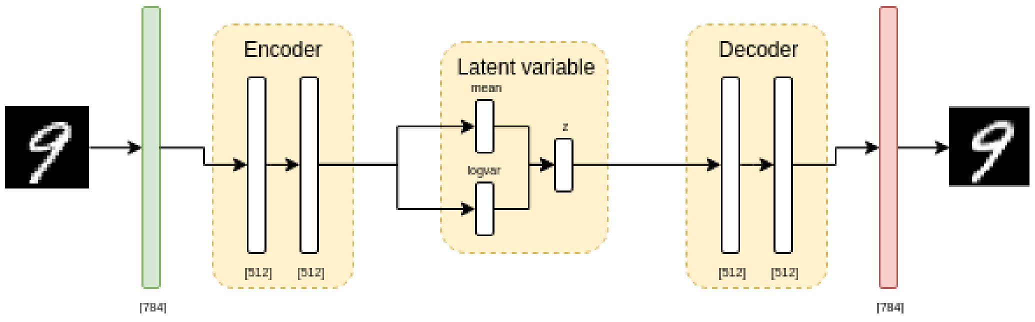


Figure 1: Architecture VAE

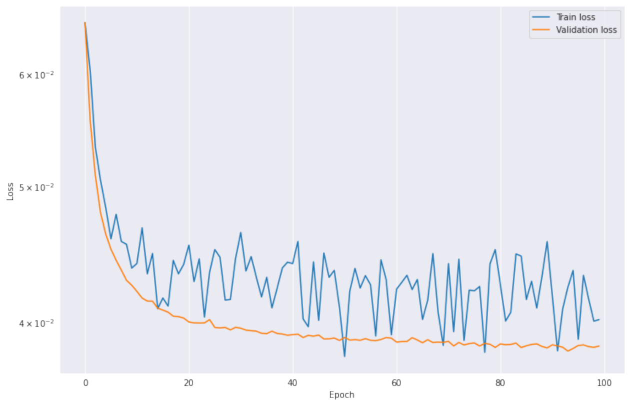
3. Results

3.1. Convolutional autoencoder

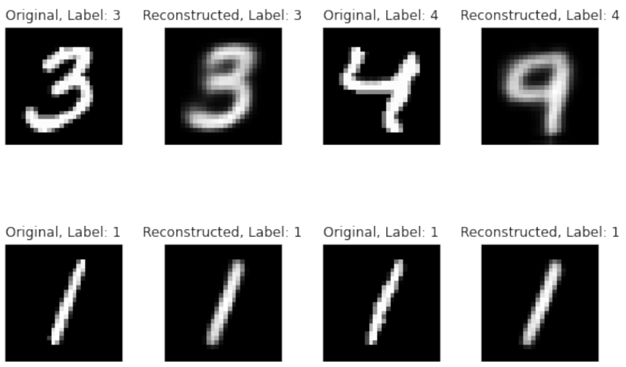
Figure 5 presents the network architecutre with the best parameters. More precisely, the best hyper-parameters founded for the model were the following:

1. channel of the first convolutional layer `conv1`: [8]
2. channel of the Second convolutional layer `conv2`: [16]
3. channel of the third convolutional layer `conv3`: [64]
4. number of neurons of the first linear layer `fc`: [64]
5. learning rate `lr`: $[1e - 3]$
6. Regularization penalty `l2`: $[1e - 5]$
7. Dropout rate `drop` :[0.1]

Figure 2a presents the evolution of the validation and the train loses per epoch. The training loss obtained for the best hyperparameters were 0.045, whereas the validation loss lead to 0.04 and the test loss 0.04. The reconstructed images for a given image can be observed in Figure 2b. As it can be seen, the model is able to reconstructed accurately the digits although blurrier than the original one.



(a) Training and validation losses evolution per epoch



(b) Reconstruction of sampled images

Figure 2

3.1.1. Latent Space Analysis

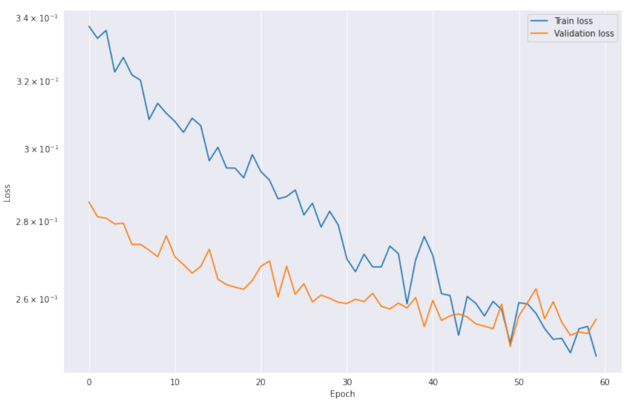
Figure 8 shows us the location of different labeled numbers within the latent space for the autoencoder with encoded dimension =2, PCA and t-SNE. In particular, it can be observed that the latents space contains gaps specially for the case for t-SNE, therefore there's not knowing on how the characters may look like in these spaces. This is equivalent to having a lack of data in a supervised learning problem, as our network has not been trained for these circumstances of the latent space. Another issue is the separability of the spaces, several of the numbers are well separated in the above figure, but there are also regions where the labeled is randomly interspersed, making it difficult to separate the unique features of characters. In particular, it can be observed that the t-SNE is the best one to cluster the digits.

Figure 6 shows the generation of new samples by sampling the classical autoencoder using a normal distribution.

3.2. Fine tuning

Figure 3 shows the evolution of the losses train and validation per epoch and the obtained confusion matrix. As it can be observed the model is able to classified correctly all the time. In fact, it was obtained a test accuracy of the model of 0.95 that is comparable with the one obtained for the first homework.

Nevertheless, it should be point out that transfer learning is a very powerful technique that allows to speed the computation by using this pre-trained network and producing high-accuracy.



(a) Training and validation losses evolution per epoch

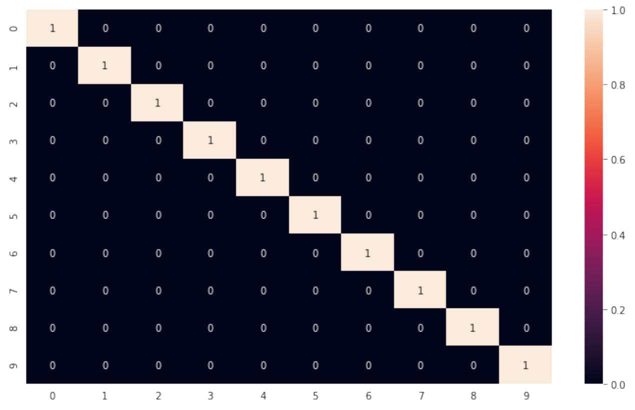
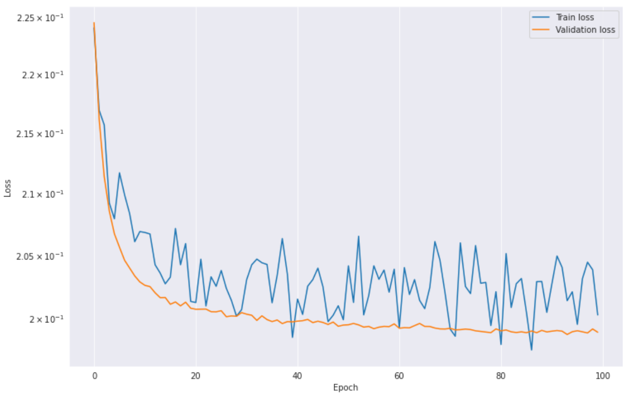

(b) Confusion matrix

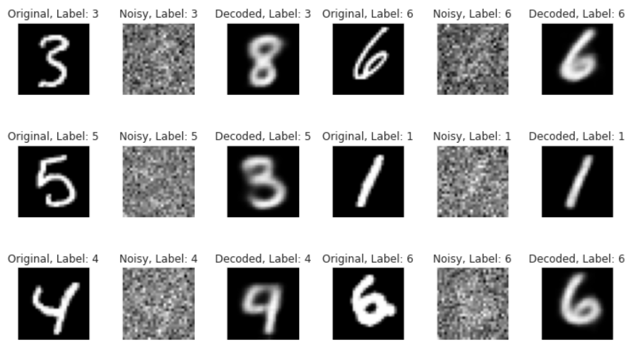
Figure 3: Results obtained by transfer method

3.3. Denoising

Figure 4a shows the time evolution of the training and the validation losses per epoch. In particular it was obtained a training loss of 0.2, a validation loss of 0.199 and a test loss of 0.199. The reconstructed images of the noisy images are shown in 4b. It can be observed that the model is able to reproduce the correct label even though the image is noisy, however for some features is not able to correctly classify. However the model is able to capture main features.



(a) Training and validation losses evolution per epoch



(b) Reconstruction of sampled images

Figure 4: Results for the denoising autoencoder

3.4. Variational Autoencoder

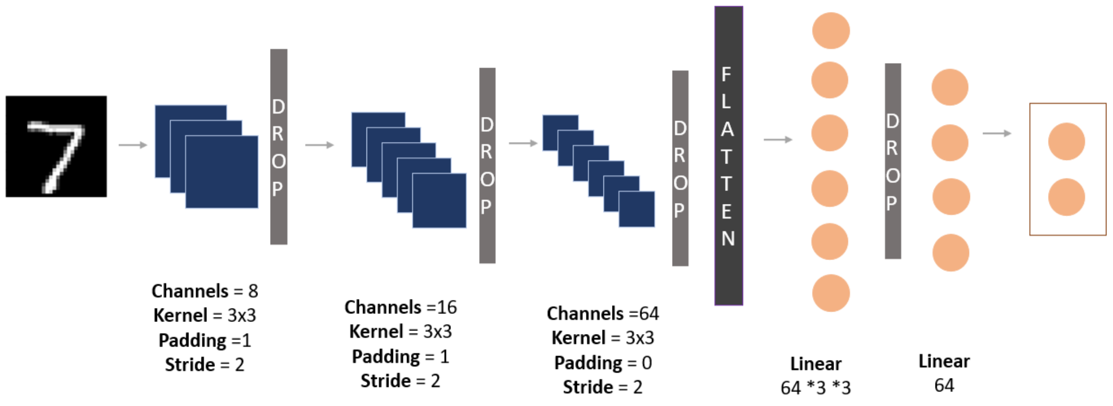
Figure 7a shows sampled reconstruction images obtained by the VAE. Altough some of the features are not perfect, the model is able to clasify correctly almost all of the features. A VAE can generate new digits by drawing latent vectors from the prior distribution.Similar to autoencoders, the manifold of latent vectors that decode to valid digits is sparser in higher-dimensional latent spaces. Figure 7b represents the 2D manifold of MNIST digits. The most fascinating thing is the smooth transitions between each of our human concepts (digits). Unlike a traditional autoencoder, which maps the input to a latent vector, a VAE maps the input data to the parameters of a probability distribution, such as the mean and variance of a Gaussian. This approach produces a structured and continuous latent space, which is useful for imaging. Even if we have only two dimension, we can already see glimpses of the learned structure. The first dimension of z encodes a bit the orientation attribute. See how generated images in the negative z1values are tilted left, while positive values are tilted right.

Figure 9 shows the presentation of digits in latent space colored according to their digit label of the vae, pca and the tsne. One should notice that the latent encoding space is filler than for the convolutional autoencoder: there is less gaps between images and there are more separate between groups than Convae. Therefore, it can be concluded that the generation of images it is much better with a VAE then the previous ConvAE, since almost all of the encoding space is recognizable. Moreover, lets recall that the VAE in this case is quite simple.

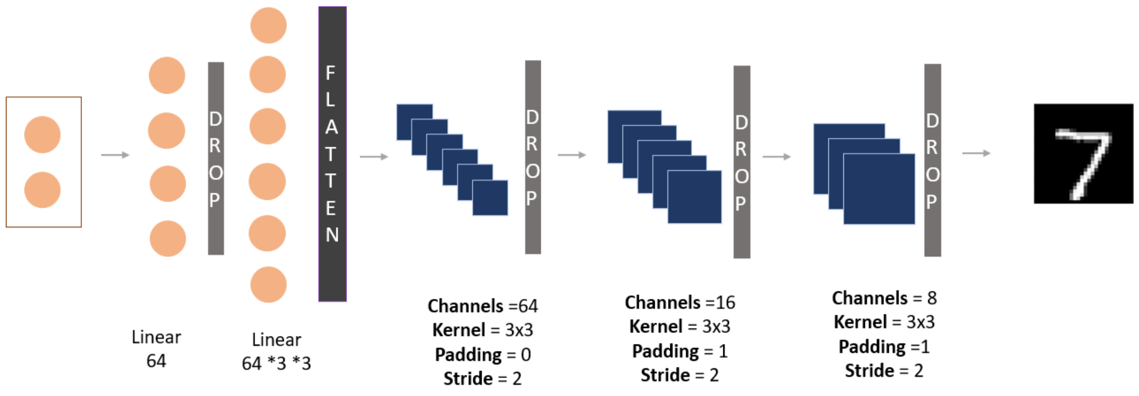
4. Conclusion

In conclusion, a convolutional autoencoder and a variational autoencoder was successfully implemented and study in the MNIST dataset. In particular, even tough not all the images were correctly classify, image denoising accuracies are indeed good. The same can be said about the supervised technique application done by applying transfer learning. Moreover, it was found that the VAE is able to generate better images that the convolutional one.

5. Appendix



(a) Architecture of the encoder



(b) Architecture of the decoder

Figure 5: Network Architecture.

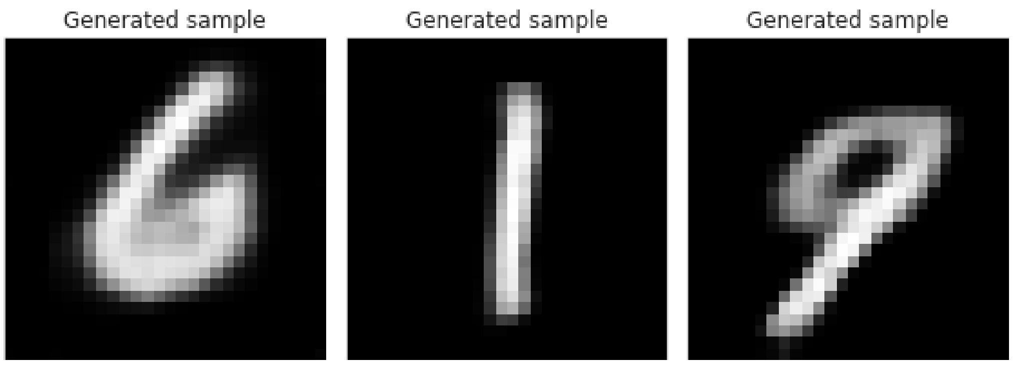
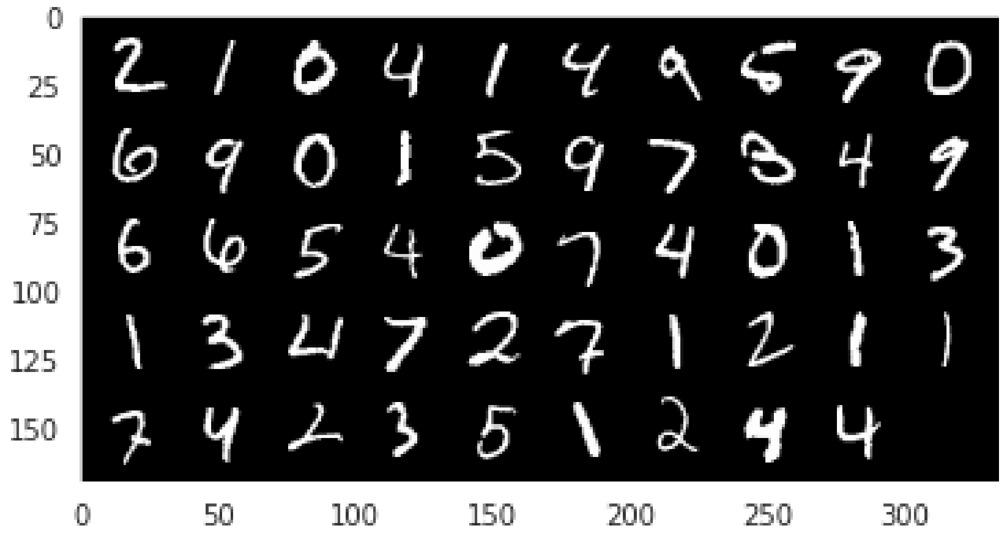


Figure 6: Generated imaged of new samples.


(a) Reconstructive images

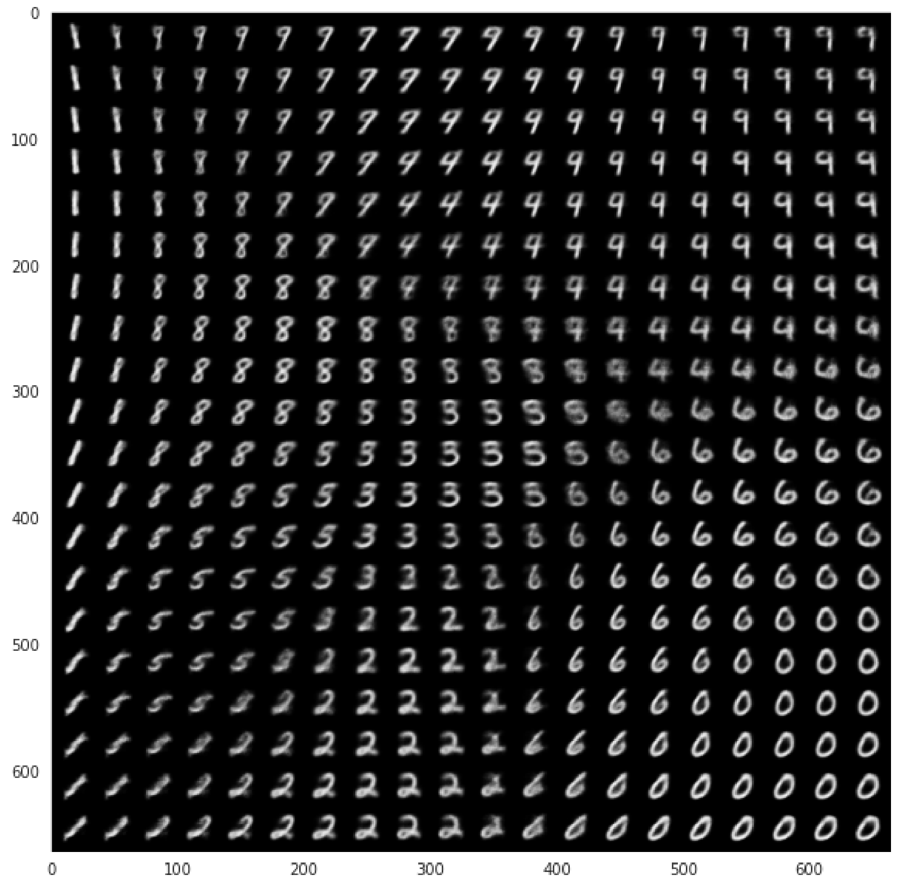
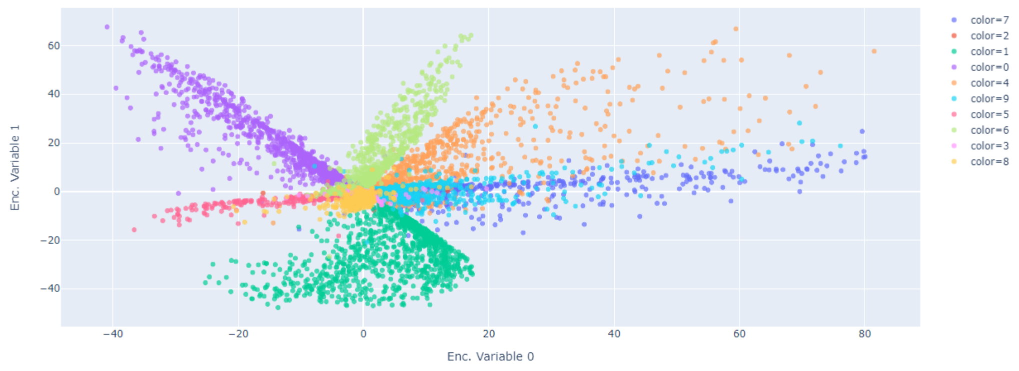
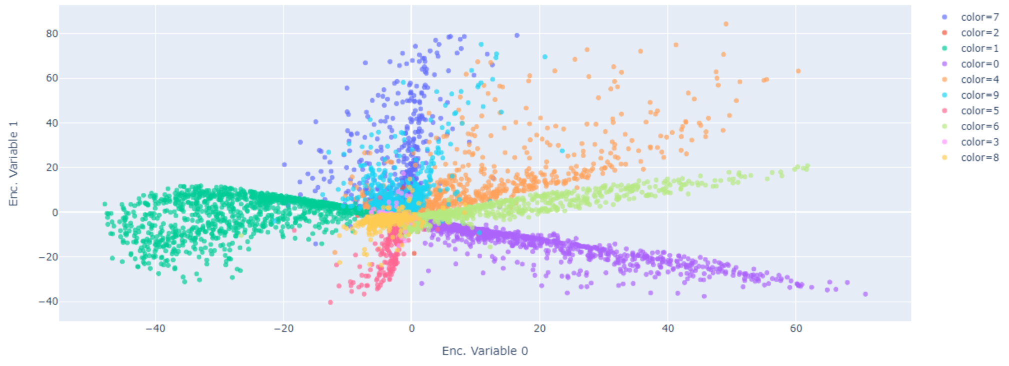

(b) 2D manifold of MNIST digits

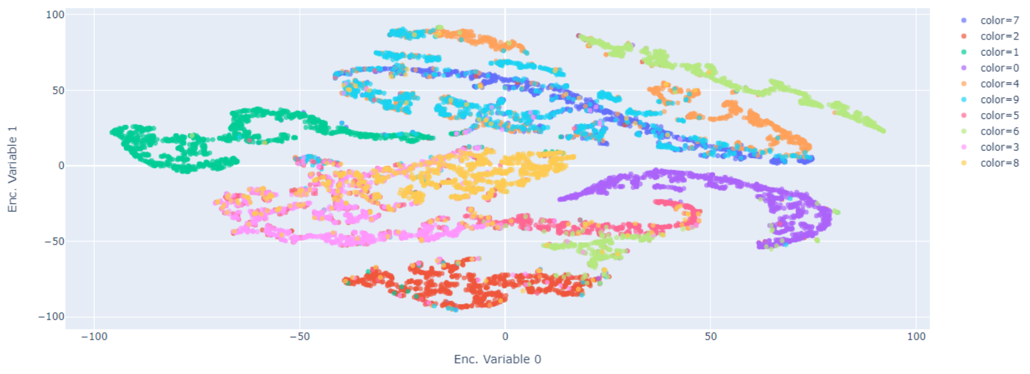
Figure 7: Results for the variational autoencoder



(a) Training and validation losses evolution per epoch

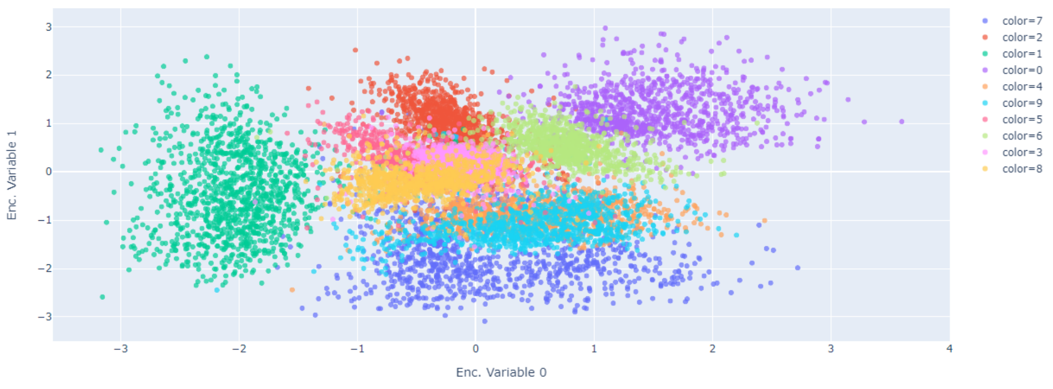


(b) Training and validation losses evolution per epoch

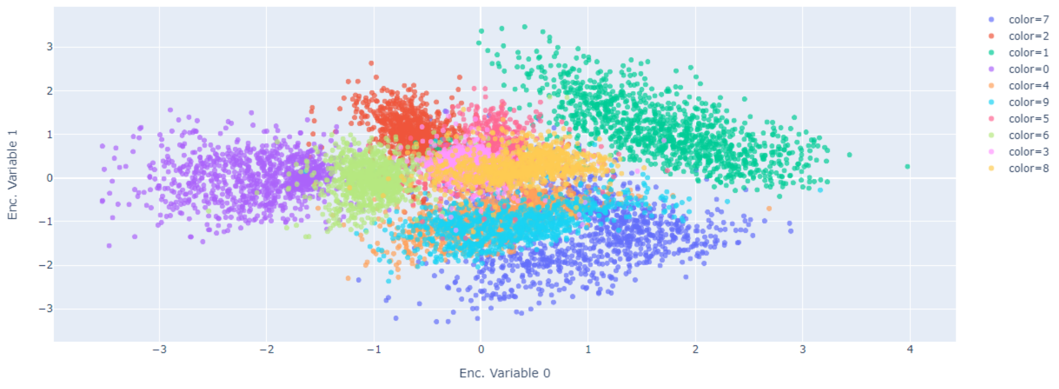


(c) Reconstruction of sampled images

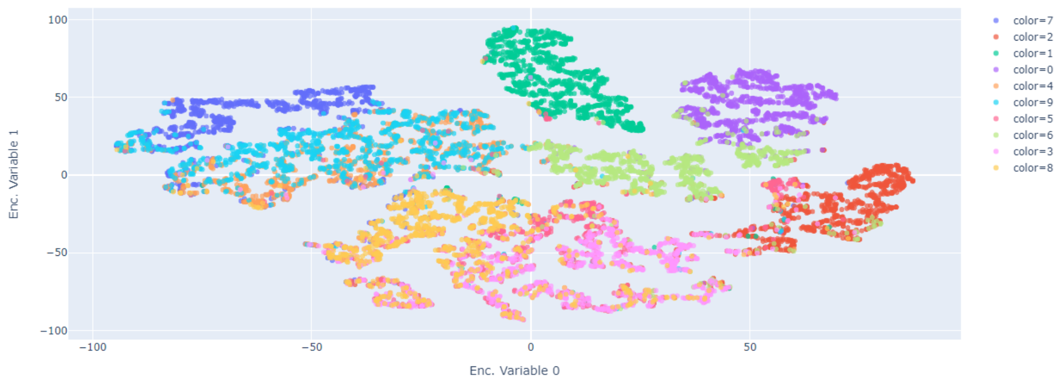
Figure 8



(a) VAE latent space



(b) PCA



(c) t-SNE

Figure 9: representation of digits in latent space colored according to their digit label