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 was done by means of a* ***Convolutional Autoencoder****. Original images were also obtained by applying a denoiser in the samples with random noise. Moreover, supervised techniques such as classification were study by applying transfer learning. Finally, a more advance architecture was investigated called Variational Autoencoder where the mapping in the latent space becomes probabilistic.*

1. Introduction

In this work, unsupervised feature learning was 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 was applied and optimize in the MNIST Dataset using PYTORCH which is an exhaustive dataset containing handwritten digits from 0 to 9 using images of size 28×28. Furthermore, imaging denoising was performed, as well as transfer learning for a classification . Finally, a more advance architecture was investigated called Variational Autoencoder.

2. Method

In this section, the models for each task are presented.

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 was composed by three convolutional layers and two linear layers, whereas the decoder was 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 was the mean square error.

Due to the fact that the MNIST dataset was quite exhaustive, a cross validation technique was not implemented. However, in order to improve the performance of our model, regularization techniques were applied. Specifically, a Ridge regression regularization (L_2) was 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 was 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 were closer together in space. Additionally, two dimensionality reduction tools were applied: Principal Component Analysis (PCA) and t-distributed stochastic neighbor embedding (t-SNE) in order to compress the dimensionality as much as possible.

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 that the convolutional layers extract general, low-level features that were 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 technique 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 were not. Furthermore, an additional layer with 10 neurons was added in order to classified the 10 digits.

2.3. Denoising autoencoder

The model architecture of the autoencoder and the hyperparameters search were 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 and VAE is the latent representation where the mapping is now probabilistic. In particular, a multivariate Normal distribution was implemented 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 was simply given by two convolutionals layers whereas the decoder by two transpose convolutionals layers. In particular, `mu` and `logvar` were standard fully connected layers that represented mean and log-variance, respectfully. Using outputs of these layers will be then used to sample latent variable z. Now, since the sampling was a discrete process, in order to allow backpropagation, **parameterization trick** was used.

As activation the sigmoid function was 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 standard deviation 1.

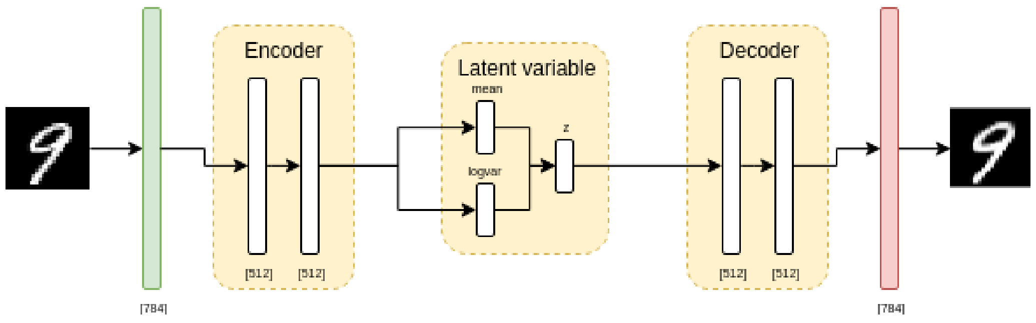


Figure 1: Architecture VAE

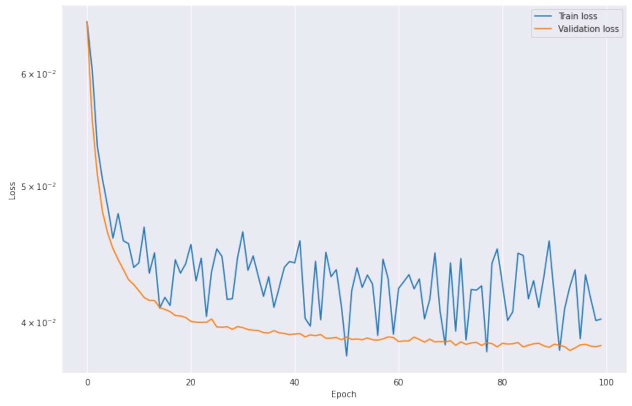
3. Results

3.1. Convolutional autoencoder

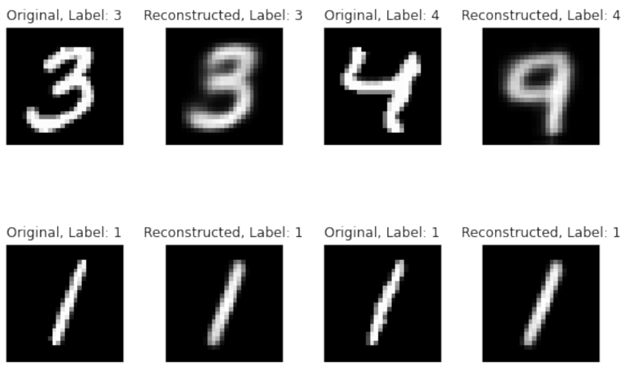
Figure 5 presents the network architecture with the best parameters. More precisely, the best hyper-parameters founded for the model were the following:

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 was 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

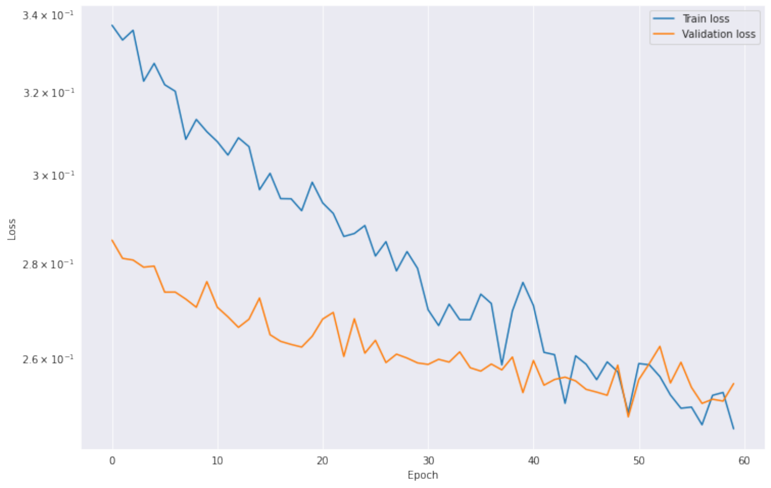
3.1.1. Latent Space Analysis

Figure 8 shows the location of different labeled numbers within the latent space for the autoencoder with encoded dimension given by two, as well as the ones obtained by PCA and t-SNE. In particular, it can be observed that the latent 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 was 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 was the separability of the spaces, several of the numbers were well separated in the above figure, but there were also regions where the labeled was randomly interspersed, making it difficult to separate the unique features of characters. In particular, it can be observed that the t-SNE was the best one to cluster the digits.

Finally, the decoder can generate new images starting from new samples. For instance if we provided a latent space of $[-40, -20]$ then the corresponding generated image by inspecting Figure 9a will be five. In particular in 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. In particular, it was obtained a test accuracy of the model of 0.95 that was comparable with the one obtained for the first homework. It should be point out that transfer learning is a very powerful technique that allows to speed up the computation by using this pre-trained network and producing high-accuracy.

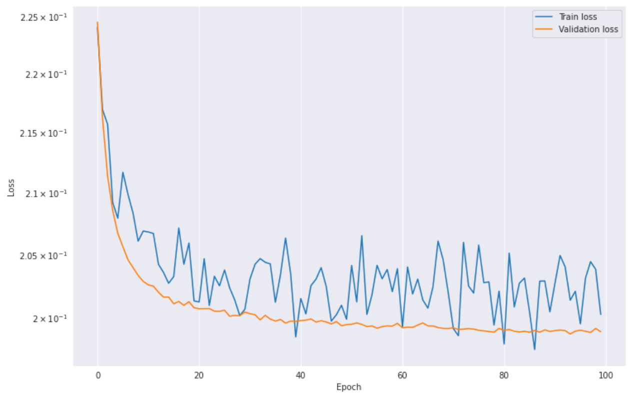


(a) Training and validation losses evolution per epoch

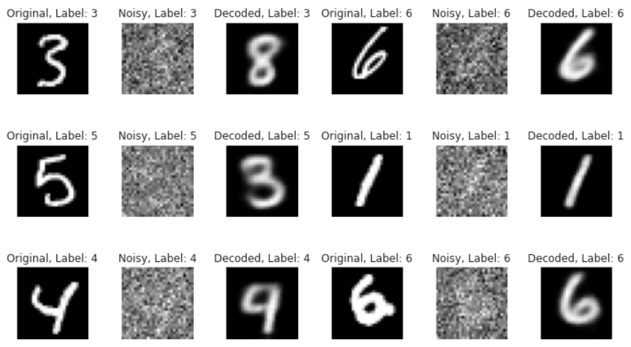
Figure 3: Results obtained by transfer method

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 was able to reproduce the correct label even though the image was noisy, however for some features was not able to correctly classify. Nevertheless, the model was 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. Although some of the features were not perfect, the model was able to classify correctly almost all of the features but blurrier than the original one. A VAE can generate new digits by drawing latent vectors from the prior distribution.

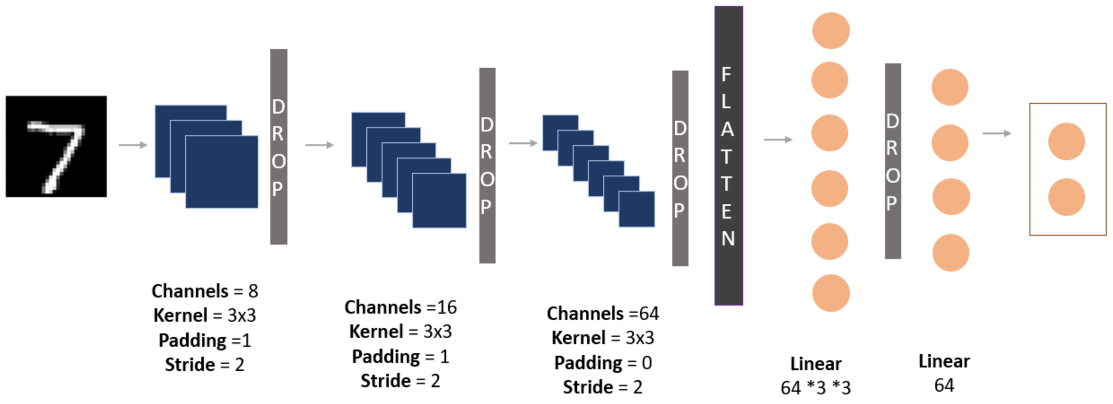
Similar to autoencoders, the manifold of latent vectors that decode to valid digits was sparser in higher-dimensional latent spaces. Figure 7b represents the 2D manifold of MNIST digits. The most fascinating thing was the smooth transitions between each of our human concepts (digits). Unlike a traditional 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 was useful for imaging. Even if we have only two dimension, we can already see glimpses of the learned structure. For instance, the first dimension of z encodes a bit the orientation attribute.

Figure 9 shows the presentation of digits in latent space colored according to their digit label of the VAE, PCA and the T-SNe. One should notice that the latent encoding space was filler than for the convolutional autoencoder (ConVAE Fig. 9a): there are less gaps between images meaning it was filler. However, altough some digits are more separate between groups, there are still a lot of superposition between groups. Lets recall that the VAE in this case was quite simple and better performances can be obtained by hyper parameter search probably leading to highly better models that the ConVAE.

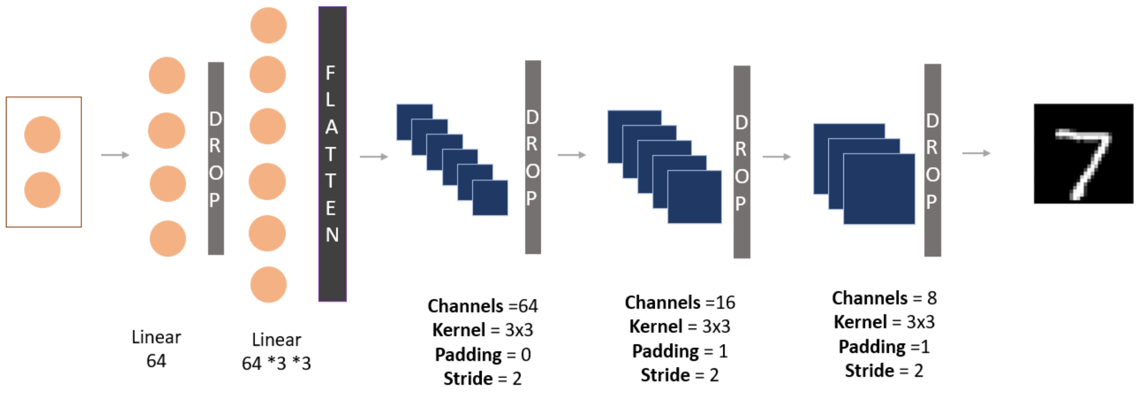
4. Conclusion

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 were indeed good. The same can be said about the supervised technique application done by applying transfer learning. Moreover, it was found that the VAE was able to fill the latent space more than the Convolutional autoencoder but still superposition between groups exist.

5. Appendix



(a) Architecture of the encoder



(b) Architecture of the decoder

Figure 5: Network Architecture.

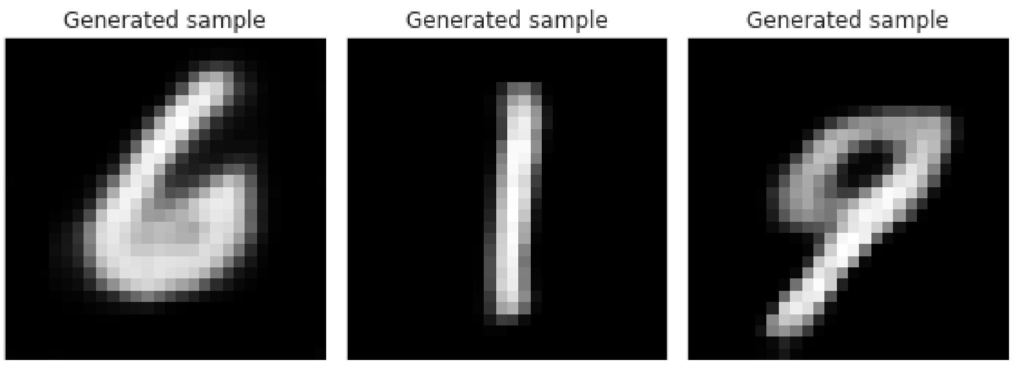
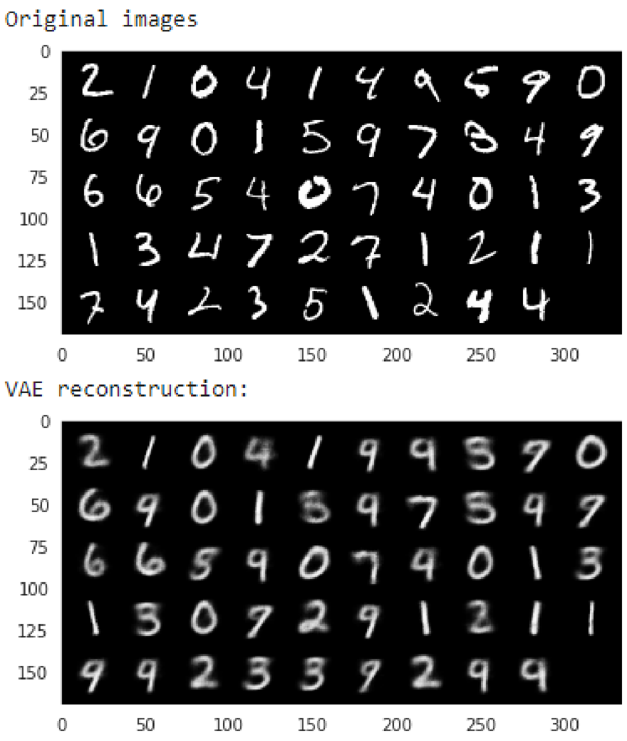
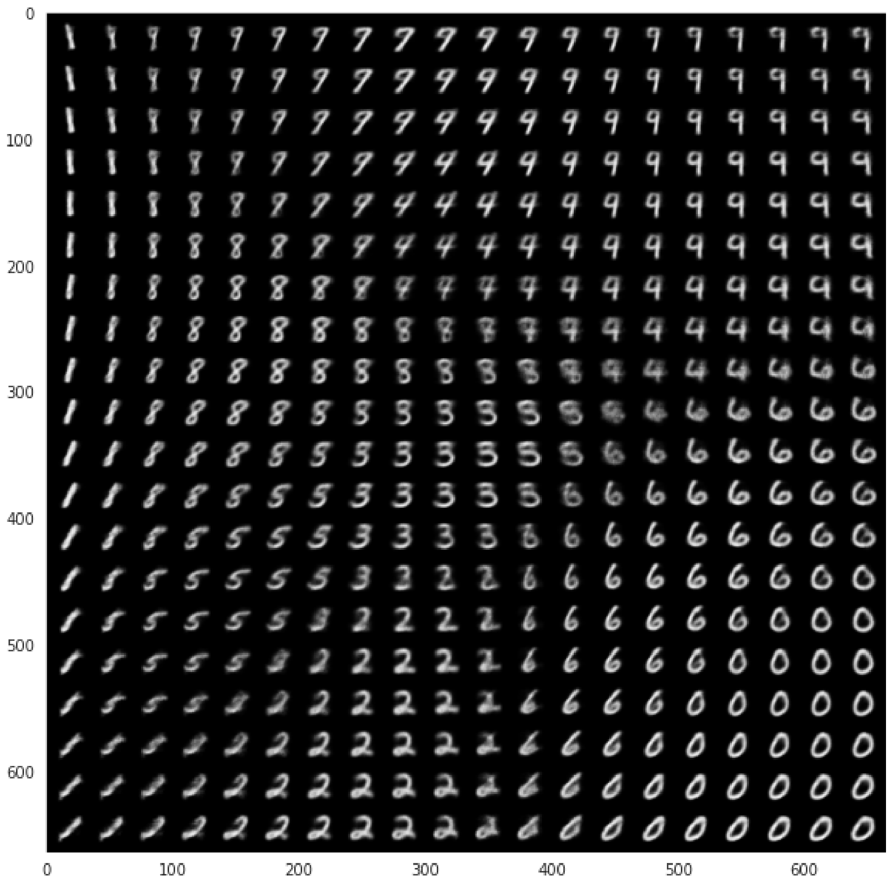


Figure 6: Generated imaged of new samples.

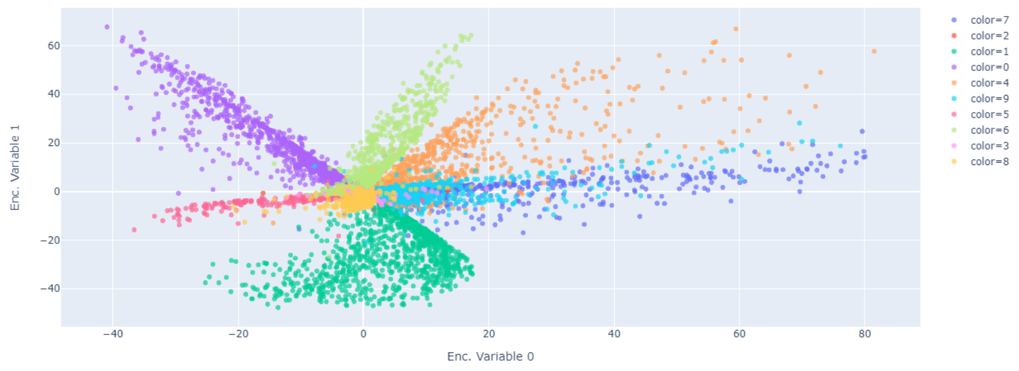


(a) Reconstructive images

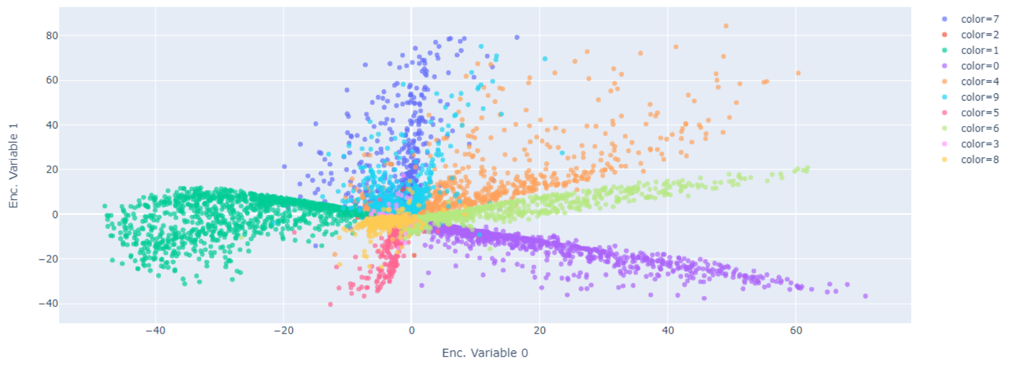


(b) 2D manifold of MNIST digits

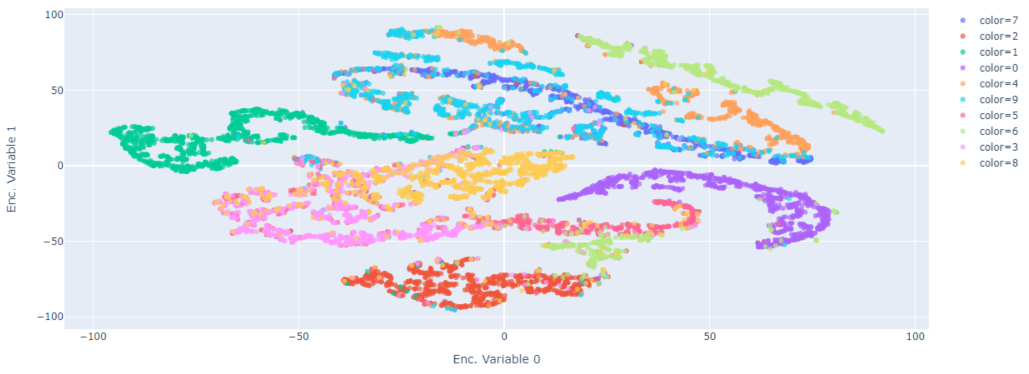
Figure 7: Results for the variational autoencoder



(a) Training and validation losses evolution per epoch

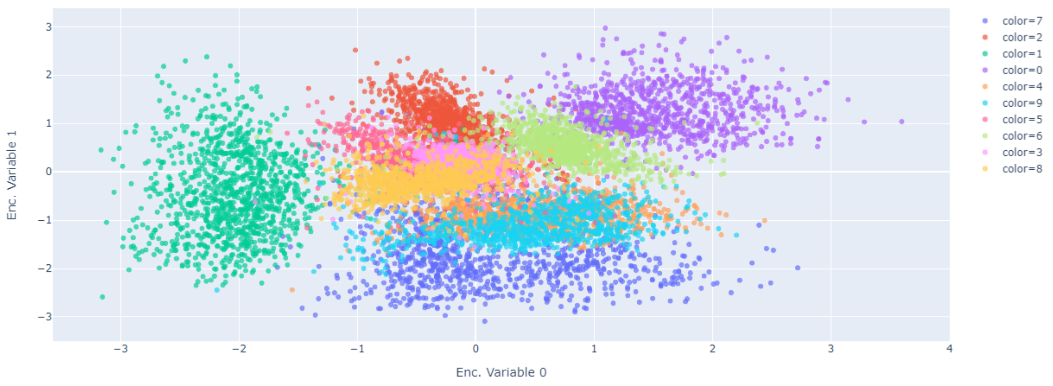


(b) Training and validation losses evolution per epoch

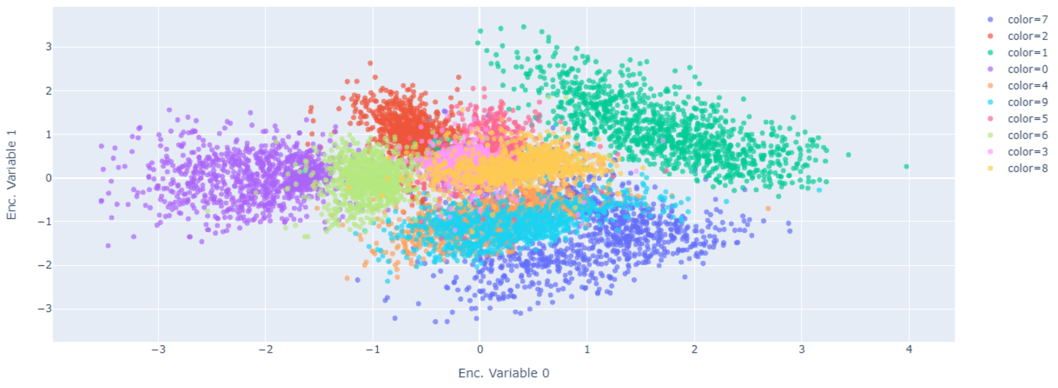


(c) Reconstruction of sampled images

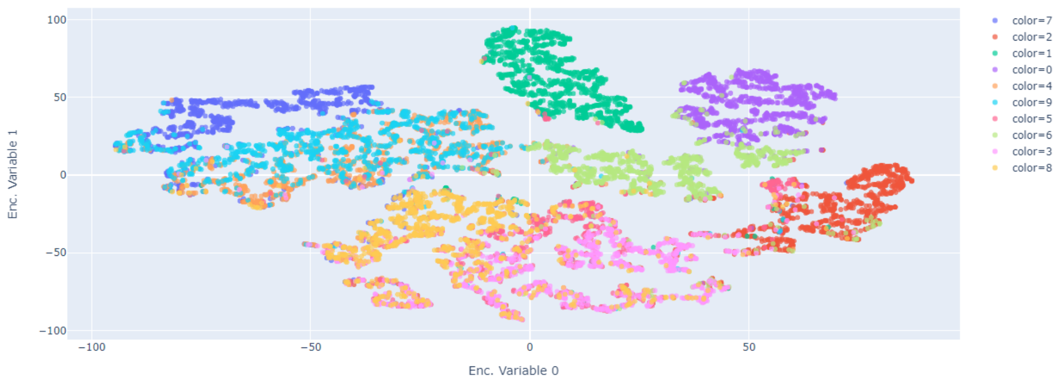
Figure 8



(a) VAE latent space



(b) PCA



(c) t-SNE

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