Università degli Studi di Padova

Neural Networks and Deep Learning

Corso di Laurea Magistrale in Information and Communication Technologies

Homework 2: Unsupervised Deep Learning

Candidato:
Filippo Dalla Zuanna, filippo.dallazuanna@studenti.unipd.it
Matricola 1238613

Anno Accademico 2020-2021

Contents

1 Introduction

This homework aims to explore and study the unsupervised learning paradigm applied to Deep Learning techniques. This paradigm consists in learning tasks using a multitude of examples to extract properties from them without a guide, so understanding some behaviour from the data itself. The data are presented pure and no labels or other information are attached to them. In this work the autoencoder is used, a model that can project the data to a smaller space and reconstruct from there. This is useful for compressing strategies or simply to extract the most relevant features from the data. Three kinds of models have been used. A convolutional autoencoder is a structure that keeps most of the properties of a standard convolutional neural network and is used to work with images. Denoising autoencoder has the same structure but this model aims to delete the noise present on the images. Finally, variational autoencoders differ on the projected space since their parameters try to approximate the data distribution over separated probability distributions. In this manner, the space would be more regularized and is possible to generate new samples from it. All these models are explored in this homework.

2 Dataset

The dataset used in this homework is the MNIST dataset of handwritten digits used in the last part of the previous homework. It is composed of 60000 training samples and 10000 test samples containing a multitude of handwritten digits examples. These are images 28X28 in black and white. The choice of the same dataset is given by the fact that some results have been compared with the ones of previous homework and changing the dataset would not have permitted it. The results would be in general applicable to other datasets of course, but some fine-tuning would be necessary for the changing of the dataset.

3 Convolutional autoencoder

3.1 Workflow

I used the Optuna framework to generate configurations of a convolutional autoencoder letting the framework find the best hyper-parameters of this model. This framework performs a random search over the parameters but uses the previous information to guide the optimization towards a better

configuration. The starting points were 12 configurations selected by me and the final goals were 40 completed. The configurations tested by Optuna framework were more since it removes by itself the ones that would not lead to an improvement. The parameters that I have been explored by the Optuna framework are:
-Optimizer: SGD with momentum, Adam or Adagrad -Learning rate: from 0.00001 to 0.01
-L2 regularization: weight decay from 0.00001 to 0.001
-Neurons of fully connected layer: from 100 to 300.
-Dimension of the encoded space: from 2 to 6.
The convolutional layers were kept of the same type and number as the best model of the previous homework to exploit the previous results. The configurations have been trained with simple validation. Cross-validation was not necessary for the same reason as the previous homework. However, the best 5 models found have been retrained with cross-validation to see if simple validation was effective. Early stopping, as previous homework, was implemented in the training that was stopped if a certain average of the validation loss is increasing. Finally, the best models have been tested and the best one is been used for the latent space study and, also, for exploit the information learned for classification purposes.

3.2 Parameters results with Optuna

The exploration with Optuna gave the possibility to cut and not train a multitude of configurations that would haven't reach good results. In this way, it could obtain better results in less time since this kind of approach is less time consuming than grid search or a lot of random configurations generated. However, this kind of method is not foolproof in every case. The relation between optimizing a loss function and the resolution of the problem is approximated and could remain stacked at a local minimum. As an example in this problem, the SGD and the Adagrad optimizers were discarded soon but we know from previous homework that SGD performed better in the test but had worse validation error, meaning that the optimization objective could have problems. So, for this reason, forcing a better exploration of some parameters could have given a better understanding of the problem. However, a very interesting piece of information taken off by Optuna was the importance of the hyper-parameters for the validation error, showing that the most important ones were the encoded dimension and the optimizer. The first is important since it represents the quantity of information that can be preserved while the second is for the training procedure. The learning rate instead is not so important, probably because of the adaptive behaviour of

the optimizers.

3.3 Cross validation comparison and test results

The cross-validation procedure compared to the simple validation one showed a minimal difference. This indicated that the simple validation set was effective. The difference between the two validation errors was less than 3% and the error histories were about identical. Also, the test results showed the effectiveness of the procedure since the test error was about the same as the validation error at a medium value of 0.018-0.019 mean square error(MSE). This error is low and so the models worked well for this compression and reconstruction task.

3.4 Latent space study

The best model owned a latent space with 6 as dimensions. This kind of space could not be easily visualized since has more dimensions than 3. For this reason, PCA and t-SNE were applied to better present the space. However, PCA presented the space as very disorganized and each category was quite superimposed because of the loss of information derived from the PCA process. The t-SNE process instead divided the space quite well but this operation is only for visualization and hasn't a function to remap back the samples. For the limits of these processes, the generation of the samples was not possible basing on them. Instead, some examples were generated using the mean of some latent space projection results of a category, showing that the numbers or the proportion between them represent the same category. The generation was done also mixing 2 categories resulting in a number that merges the two and not generates something very different, an interesting aspect that shows a good division of the latent space among the various categories.

3.5 Exploit encoder for classification

The information learned during unsupervised problems could be exploited to deal with supervised ones. In this case, the information learned from the convolutional encoder part could be used as general good feature extraction from the data. For this reason, the parameters of the convolutional part were fixed and only the fully connected part was made trainable again, changing also the latent space to a layer for classification purposes. The training was made with a maximum of 30 epochs with early stopping. This took about two minutes and gives an accuracy of 97,56% that is very similar to

the performance taken from the previous homework but needs a much short training process(8 minutes in previous homework), showing the effectiveness of the previously learned features.

4 Denoising autoencoder

4.1 Workflow

For the denoising autoencoder, the model taken was the same as the best found in the convolutional autoencoder part. The work in denoising autoencoder is practically the same as previous but with modified data. I added Gaussian noise by adding numbers generated from a normal multiplied by a factor to keep the amount of noise acceptable. The labels, instead, were the original images. In this manner, we had the dataset for the training of the denoising autoencoder, which was done as previous with the usual early stopping. Then it was tested on the test set with the same noise added to the training set and with other noises.

4.2 Test results

The model reached test results of 0.0188 MSE, which is comparable with the result of the convolutional autoencoder, meaning that the model is capable to manage the noise added.
The tests have been done also with more intense added noise and the results were:
-Noise 2 times as original: 0.0201 MSE.
-Noise 3 times as original: 0.0261 MSE.
-Noise 4 times as original: 0.0381 MSE.
-Noise 5 times as original: 0.0525 MSE.
The network behaves well even with very big noise, retrieving the base shape. This worked also with a different kind of noise, reconstructing the base shape with salt and pepper noise. So it shows generalization properties over this problem.

5 Variational autoencoder

5.1 Workflow

The model has been modified in the encoder part while the decoder remains the same. The latent space in variational autoencoders is represented by the

mean and the variance parameters of a multi normal distribution. In the training phase, the code to be reconstructed is not taken directly from the output of the encoder but is sampled according to the resulting parameters, in a way to tune these distributions. In the evaluation phase, only the mean is taken into account. Then this model is trained and tested with the same configuration of the best convolutional autoencoder found before. This was done with the usual training procedure, apart from the different loss function(KL divergence loss) that had to be changed for this model. The latent space was analyzed with the focus of seeing if it was more regular than the convolutional autoencoder. For this reason, also a model with 2 as latent space dimension was trained and tested since the visualization didn't require any map process.

5.2 Test results and latent space study

The test results were in line with expectations since the performance are the same as the convolutional autoencoder. The ideal regularization of the space didn't deteriorate the performances that had 0.0205 as MSE.
Regarding the latent space, PCA and t-SNE were applied to better present it, since the best model owned a latent space with 6 as dimension like before. In this case, PCA presents the results of the regularization induced in this kind of model, since all the values are confined and limited. However, the process superimposed the classes with the compression. The t-SNE process divided the space quite well but had the same limitations as told before. The generation in this case was done only to exploit the regularity of the space. A grid of 2D samples was generated and projected with the inverse of the PCA. This brought to see that generation from the space changed smoothly with the change of the parameters and there isn't a sharp change between classes.
The model with 2 as latent space dimension had 0.0373 MSE error as test performance. This is worse than the other model but the interesting part is the representation of the data in the latent space. This is well organized in clusters and, even if there are overlaps, they regarding similar numbers meaning that the model identifies similarities, an interesting generalization property. Probably a model with better performance would divide and organize better the space. The generation made with the 2D grid gave the same behaviour as before.

A Convolutional autoencoder appendix

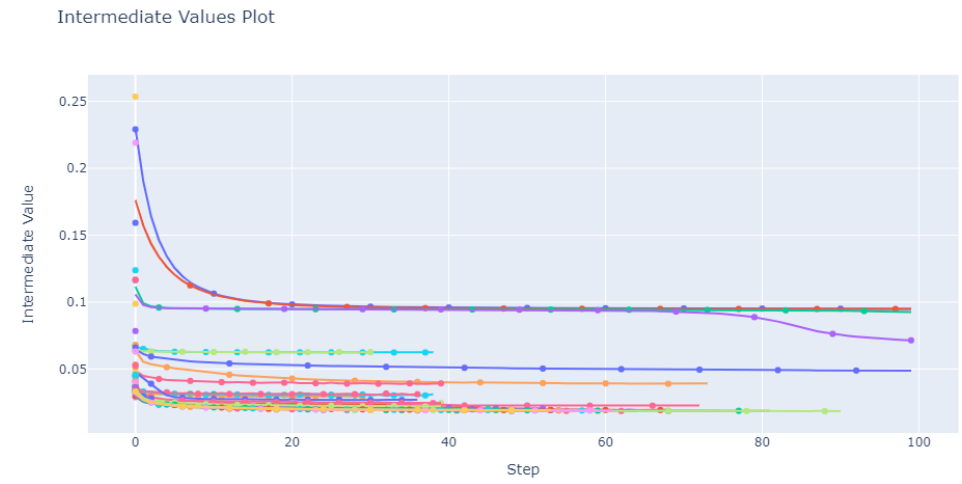

Figure 1: Optimization history of Optuna configurations

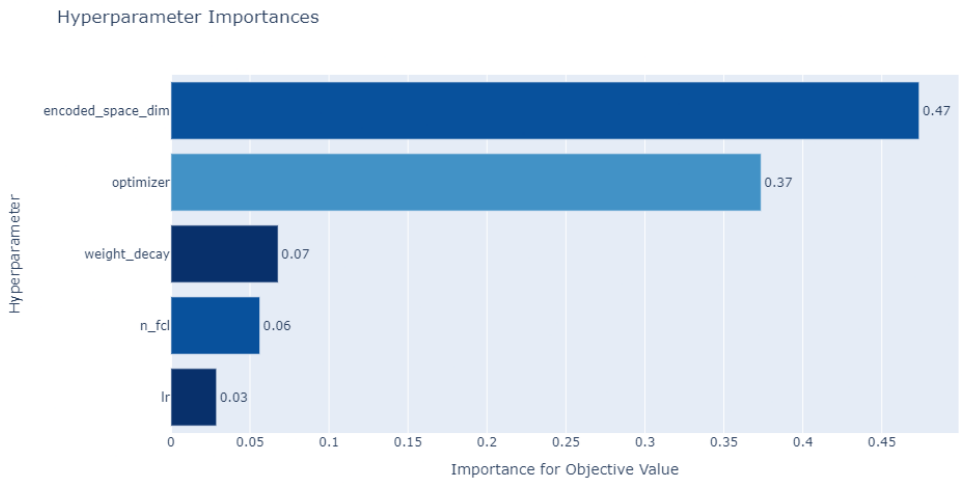

Figure 2: Hyperparameters importance in optimization

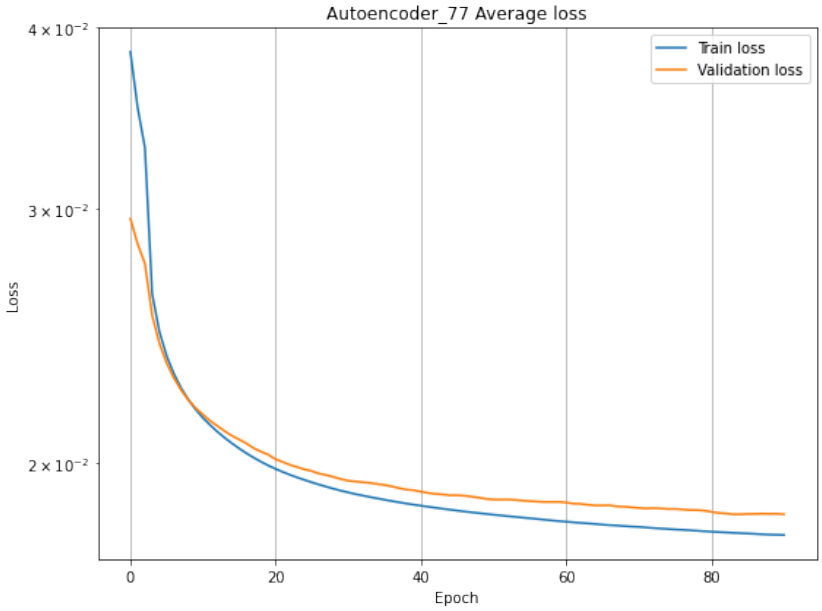

Figure 3: Training history of best model

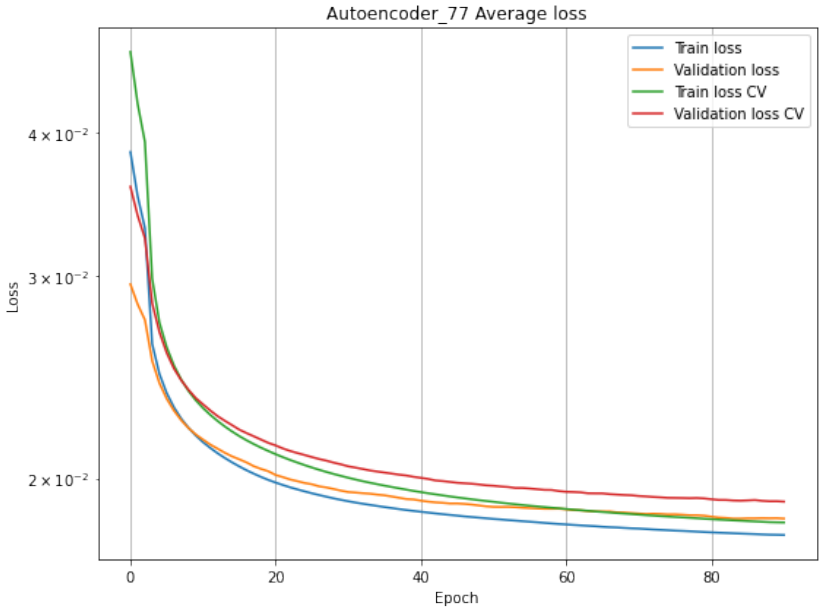

Figure 4: Training history comparison with and without cross validation

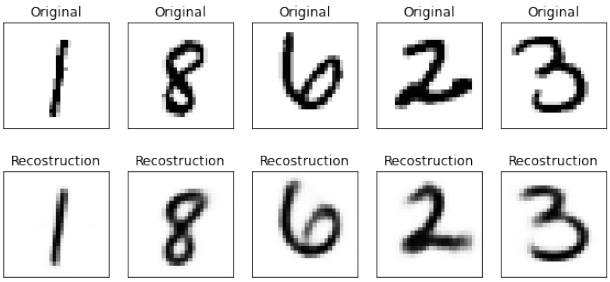

Figure 5: Convolutional autoencoder samples reconstruction

```
Enc. Variable 0    28.336331
Enc. Variable 1     1.519525
Enc. Variable 2    -4.319949
Enc. Variable 3   -13.529023
Enc. Variable 4   -13.910025
Enc. Variable 5    -0.169529
dtype: float64
Sampled image, class 9
```

```
Enc. Variable 0     0.505900
Enc. Variable 1     2.045137
Enc. Variable 2     9.832751
Enc. Variable 3   -23.877063
Enc. Variable 4     1.326569
Enc. Variable 5    14.995889
dtype: float64
Mixed sampled image, class 3 and 8
```

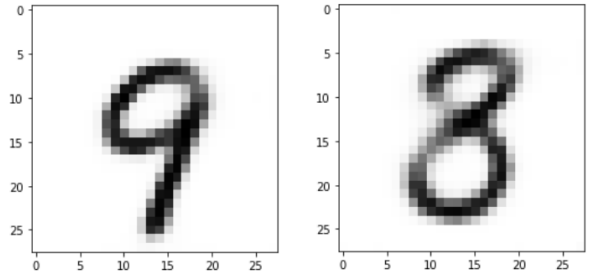

Figure 6: Convolutional autoencoder samples generation

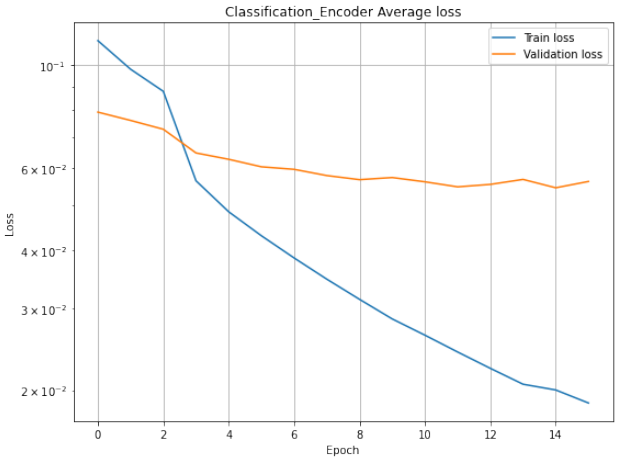

Figure 7: Classification model training history

B Denoising autoencoder appendix

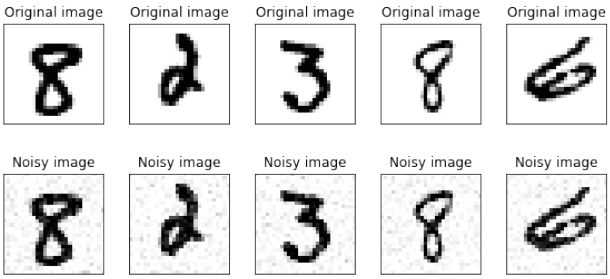

Figure 8: Dataset with noise

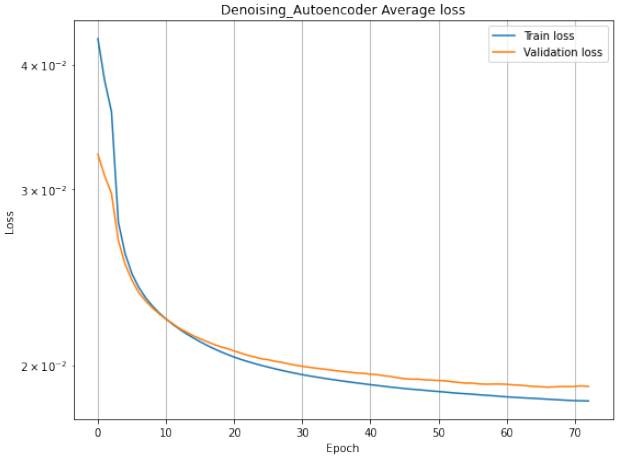

Figure 9: Denoising autoencoder training history

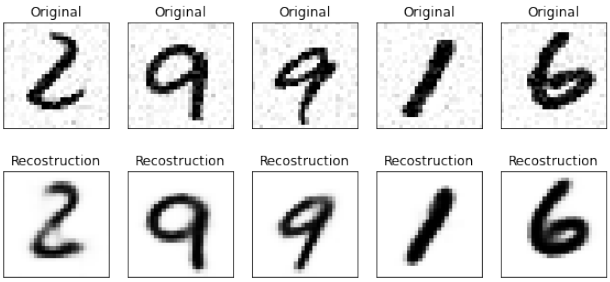

Figure 10: Samples reconstruction from test set

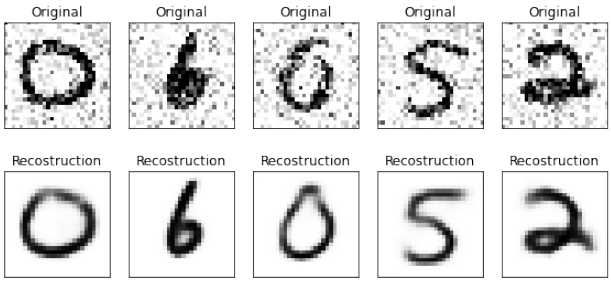

Figure 11: Samples reconstruction with three times bigger noise

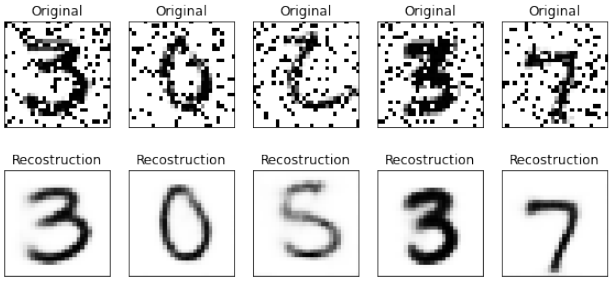

Figure 12: Samples reconstruction with salt and pepper noise

C Variational autoencoder appendix

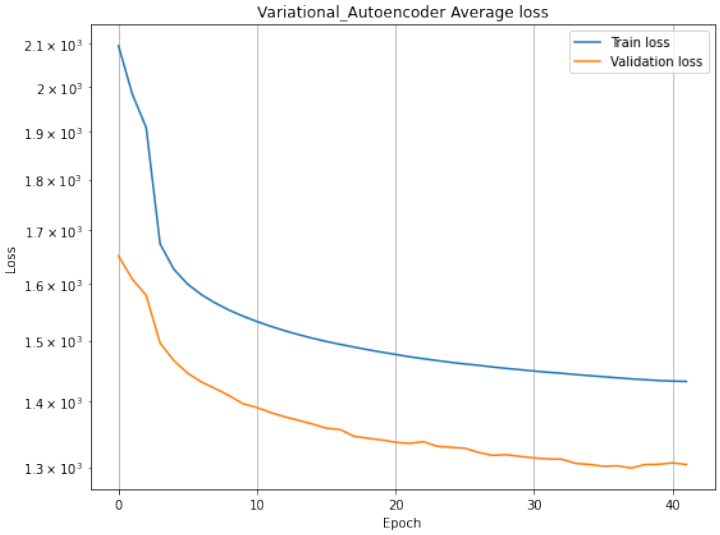

Figure 13: Variational autoencoder training history

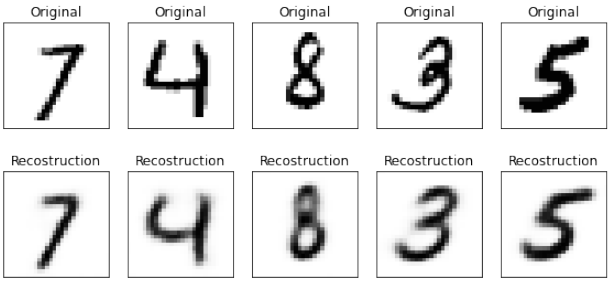

Figure 14: Samples reconstruction from test set

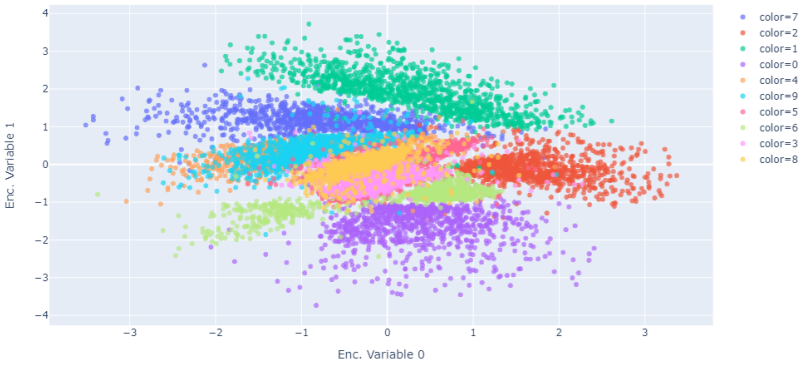

Figure 15: Latent space of model with 2 as space dimension

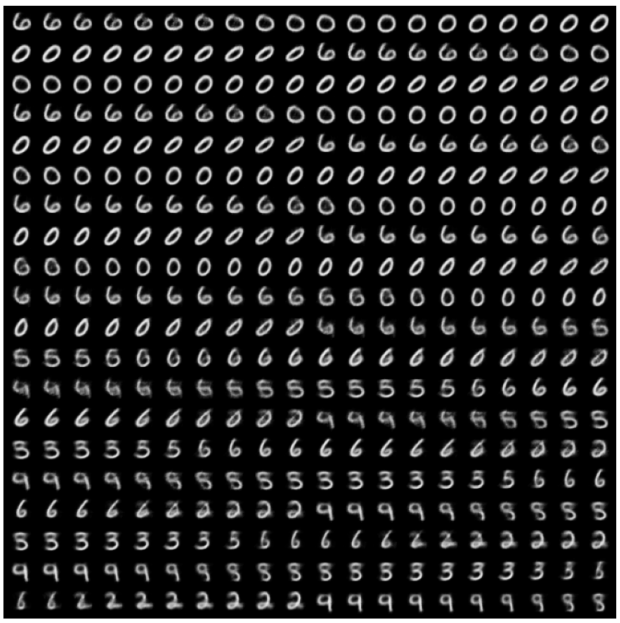

Figure 16: Samples generated from latent space