# Homework 2 - Unsupervised Deep Learning

Simionato Giuseppe (2029013)

Academic Year 2021-2022

## 1 Introduction

### Image reconstruction and generation

In every day life, most of the data are unlabelled. The unsupervised learning is a machine learning technique that uses algorithms to analyze and cluster unlabeled datasets. These algorithms are able to discover hidden patterns or grouping data without the need for human intervention. Its ability to discover similarities and differences in information make it the ideal solution for exploratory data analysis, but also they are powerful due to their generalization ability which can be applied to solve more easy supervised tasks.

In this work, it is presented the implementation and test neural network models for solving unsupervised problems. Firstly is studied a convolutional **AutoEncoder (AE)** model to see how this architecture is able to build an abstract representation of the data and how it is able to reconstruct a given image without any supervisor. It is also explored the latent space of the autoencoder to understand how it has constructed the representation of the samples and if it can be extracted some information about that.

The second unsupervised model used in this work is a $\beta-$**Variational AutoEncoder (**$\beta-$**VAE)** which basically extend the idea of the AE where instead of trying to reconstruct deterministically the input image, now the encoder returns a distribution over the latent space. From this distribution it is sampled a representation in the latent space and starting from this representation the decoder reconstruct the image.

Finally, it is implemented a **Generative Adversarial Network (GAN)** where the unsupervised task is mixed with the supervised one in order to obtain a model able to generate new samples similar to the real ones. This model is compared with the previous ones in the generative task in order to find the architecture which can reconstruct the input images with the best performance.

The first two models are also trained with a latent space dimension equal to two in order to test the ability to encode the data of these architectures compared with the widely used techniques of PCA and t-SNE.

### Fine-tuning

The last part of this work is focused on a supervised problem: the **classification of images.** Starting from a given model previously trained in the same or in another task, it is unfreezed the entire model and re-trained it on the new data with a very low learning rate. This can potentially achieve meaningful improvements, by incrementally adapting the pretrained features to the new data. This procedure is made using the previously mentioned autoencoder model, but also using the ResNet 18 architecture provided by torch. This last model is fine-tuned with a pre-trained model and with random weights to see the benefits of having an architecture already trained, even if with another dataset.

In the following, every model is trained and tested using the images of FashionMNIST dataset. As for the previous homework, you 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.

**Regularization methods:** it has been implemented some regularization methods and it is tested the impacts of these procedures on the final performance. The methods are: batch normalization and dropout layers, the advanced optimizer Adam, the L1 and the L2 regularizers implemented separately.

# 2 Methods

## Dataset Analysis

The pre-processing of the dataset is simply composed by some transformations. After loading the train and test dataset, to the first one are applied soem simple transformations: the samples are flipped horizontally with a certain probability and then they are converted into Tensors applying also a normalization to each image to obtain a lighter input to make the model converge faster. Indeed, when the data is not normalized, the shared weights of the network have different calibrations for different features, which can make the loss function to converge very slowly and ineffectively. The same transformations are applied also to the test set except for the flipping. The training set is then splitted in two parts: one for the training (80% of the whole dataset) and one for the validation (20% of the whole dataset).

**Fine-tuning pre-processing:** the dataset for this second task is manipulated in a different way. They are used the same transformations of the classification problem used in the homework 1: the images are randomly rotated and vertically flipped, then they are converted to tensors and normalized. For the ResNet model, beacuse of this architecture needs a specific input, the samples are also converted from a one channel image to a three channels one by simply repeating the gray scale channel three times.

## Autoencoder

The architecture of the AE model is made of two main parts: the encoder and the decoder, separatly declared in different classes to archieve more flexibility. The **encoder** is composed by two convolutional blocks, each of them composed by a convolutional layer, the ReLU activation function, the batch normalization layer and a max pooling. After the convolutional part there is a linear layer, followed by the dropout and the ReLU and finally there is a last linear layer with the number of filters which correspond to the latent space dimension. The **decoder** has firstly a linear layer, followed by the dropout and the ReLU activation, and then there are two deconvolutional blocks made of a convolutional transpose layer and a ReLU function. The last layer one is another convolutional transpose and correspond to the output layer which gives in output an image of the same dimensions of the input of the encoder.

The total number of trainable parameters is: 29 716. The evaluation of the model is done in two training sessions: one before the hyperparameters optimization and one after. The first training is done to see how the architecture is simply able to reconstruct the images and to also see if everything works properly. The final training is done with the **best hyperparameters** and using the **k-fold cross validation** (k=5) to obtain a better evaluation of the performances and more stable results. The hyperparameters optimization is done with Optuna over 20 trials to find the best values of: the latent space dimension, the batch size, the probability in the dropout, the learning rate and the weight decay. Finally, it is studied the interpretability of this model and how it is reconstructing the samples. This is done looking at the filters of the different layers, looking at the activation for a specific sample and exploring the latent space plotting two dimensions and using the PCA or the t-SNE to represent them in a unique 2d plot.

For this model and for the $\beta-$VAE is used the Mean Squared Error (MSE) loss between the input image and the output one in order to minimize the difference and obtain the best reconstruction. Notice that in the second model it is added to the loss function a regularisation term to force the latent distribution to be as close as possible to a standard normal distribution.

### $\beta$-Variational Autoencoder

The VAE model is similar to the AE one, there are the same two convolutional blocks for the encoder, then there is the latent space made of two linear layers which represent the mean and the standard deviation of the latent distribution. The decoder is simply composed by two convolutional transpose layers. The main feature of the $\beta$-VAE is the presence of a penalization term in the KL divergence using the hyperparameter beta that balances the latent channel capacity and independence constraints with reconstruction accuracy. With a larger beta the representations are more disentangled. The hyperparameters are optimized experimentally and for the training are used the best ones.

The latent space is then explored. For each dimension, starting from a point of the latent space, are sampled some images keeping fixed all the dimensions except one and moving in the latent space along the chosen one. They are also plotted two dimension in a 2d plot and then are explored these dimensions at the same time, keeping fixed all the others.

These first two models after the training with the best hyperparameters and after the exploration of the latent spaces are re trained with latent space dimension equal to two. This is done to test the ability of encoding data of these models and to see how they are able to keep only the relevant information. Indeed, the possibility to compress high dimensional datasets in few dimensions and represent them in a plot is really important in many fields. These two encoed representations obtained with the AE and the VAE are compared with the representations of the dataset with the two most famous techniques, PCA and t-SNE.

### Generative Adversarial Network

The GAN is a modern approach intruduced recently and it is able, if properly trained, to generate new samples using only forward propagation thanks to the absence of the encoder part. The implemented architecture is made by a **Generator** with a linear layer which takes in input a random pattern noise of 32 elements. This dimension is found to be the optimal after some trials. After this first layer, there are three deconvolutional layers with the corresponding activation functions. The **Discriminator** is composed by two convolutional layers and a linear one which provide a label (true image or false image), each layer is followed by an activation function. For the optimization are declared two different optimizers with the same settings, the Adam with learning rate=0.0002 and beta1=0.5. As loss function is used the Binary Cross Entropy. From the DCGAN paper, the authors specify that all model weights shall be randomly initialized from a Normal distribution with mean=0, stdev=0.02 and so the weights are initialized the same way also in this work ([1]).

It is important to note that this GAN model is very simple and it has no claim to achieve high performance, but it is only implemented to see what are the main features of this machine learning technique and to compare it with the sample generation ability of the autoencoder-based architectures.

### Fine-tuning

As already mentioned, firstly it is fine-tune the **autoencoder** architecture previously trained to solve the classification task of the same dataset. It is kept the encoder part with the same weights and it is connected to a linear layer with number of filters the same as the number of classes of the FashionMNIST dataset. The new obatined architecture is trained and tested with the same hyperparameters obtained with the previous optimization, except for the optimizer, which now has a learning rate of 0.0001.

The second fine-tuning is done using the **ResNet 18** architecture. This model is loaded twice, the first one with the pre-trained parameters and the second one with random weights. To the two loaded architectures is added a linear layer as for the fine-tuning of the autoencoder. Due to the high number of parameters of this model, it is decided to train only a part of them to avoid seed up the training. In particular, only the last four layers are trained.

Finally, it is declared the model with best best hyperparameters of the **homework 1** (a CNN) and the weights of the best fold of the cross validation are loaded. The so declared architecture is only tested and not trained in this work.

# 3 Results

## AE results

For the **first training** is used a latent space dimension of 100. The test results after the two training are reported in the Table 1. It can be seen that the reconstruction of the image after the first training, Figure 1, and after the optimization of the hyperparameters (Figure 2) is different, indeed thanks to the best hyperparameters the model is able to gives in output a more detailed image. The pixel of the borders are more defined and more details are present.

| | **First training** | **Final training** |
|---|---|---|
| Loss | 0.023 | 0.013 |

Table 1: Autoencoder results (for the final trining is reported the average loss over the 5 folds)

The **best hyperparameters** are: latent_space_dim= 38, batch_size= 48, p= 0.0, lr= 0.002, weight_decay= $5 \cdot 10^{-5}$.

**Loss trend:** looking at the trends of the final training in Figure 3, we can see that both the training and validation values are always decreasing even if there are some oscillations and it has not reached any stationary state. We can affirm that the model is not overfitting our data, that the architecture is able to learn properly. Moreover, we can also notice that it is decreasing at the end of the training which is of paramount importance if we are looking for the minimum of the loss. It is reported only the one one the last fold because they have more or less the same trend.

**Filters visualization and layer's response:** if we look in more detail how the network works after the final training, we notice that the representation of the filters of the encoder layers (Figures 7 and 8) but also for the decoder layers (Figures 9, 10 and 11) is the same of some common filters used in computer vision, we can recognize for example the horizontal and vertical edge detectors. This is not surprising if we know that the filter operation is a convolution and so the convolutional layers must act at the same way. It is more interesting the fact that the encoder filters operates in order to extract information about the structure of the data while the decoder filters are trying to extract information to reconstruct again the image starting from a compressed representation. Another interesting thing to notice is the response of the different layers for a given image, in this case a pullover. Starting from the input, it is clear the increasing level of abstraction in the response of the network, as it is clear from the Figure 12 and 13: the encoder is keeping only the relevant information while it is discarding what is not important for the latent representation. On the other hand, the decoder gives a more defined image nearer is the layer to the output (Figure 14 and 15): it is confirmed that the model is not learning by hard the images and it is not overfitting the training set, it is learning something about the latent structure of this dataset.

**Latent space visualization:** the plot of two encoding dimension is reported in 4. Because of the latent space dimension is grater than 2, in order to visualize the whole space we have to operate a dimensional reduction through PCA and t-SNE. The plots are reported in Figure 5 and 6, where in particular using the second technique the points of different classes forms clusters even if some clusters are mixed together, this is probably due to the fact that they are similar classes, like shirt and t-shirt.

## $\beta$-VAE results

The results for this second model is the same of the simple autoencoder, but as it is highlighted from the loss in Figure 17 and from the reconstructed images in Figure 16, the results of the $\beta-$VAE are more robust. First of all, the best results are obtained with a latent space dimension of 18, which is smaller than the 38 dimensions of autoencoder. Moreover, the final loss of this model is 0.014, but the trend has not oscillations and the results of the training and validation are really similar: this suggest a very stable training and a good generalization of the model. The reconstructed images are more realistic than the ones of the autoencoder and it is really easy to confuse them with an image of the dataset.

**Latent space visualization and exploration:** the plot of the latent space is reported for two dimensions in 18 while in Figure 19 and 20 are plotted the representations of the latent space using PCA and t-SNE. As for the first model, the representations highlight a structure that the latent space is able to discover, indeed in particular fro the t-SNE representation and more in this model than in the previous one, there is a clear hierarchical structure where we can see the data with different labels are far from each other and data with the same labels forms a cluster. This is really interesting and confirms the good performance of the model: without any supervision and without any label the network discovered a structure in the dataset and it has learnt that structure.

After looking at the latent space, it has been explored along the different dimensions. Here are reported the images obtained starting from a point in the latent space and decoding that point in an image: keeping fixed a all the dimension except one we are able to see the features of that dimension and what it is encoding. The Figure 21 and 22 show the result of the exploration along the dimenson number 7 and 16. It is also reported the exploration along these two dimension at the same time while all the others were fixed, see Figure 23. In all these cases the chosen starting point corresponds with the original image of a bag.

## GAN results

The results of the GAN are reported in Table 2. After 100 epochs the model has not learnt how to generate samples similar to the real ones. The results of the loss, even if the generator has a smaller value, are not good, in fact the trend in Figure 24 highlights some problems. The staring trend is the typical one of the GANs, where initially the two loss has the opposite trend of the optimal one: at the beginning the discriminator is able to tell easily whether some data is fake. However, the generator learn very fast how to generate samples that the discriminator is not able to recognize reaching rapidly a stationary state with little oscillations. A possible explanation of this results is connected to the vanishing gradient, indeed the best results are at an equilibrium state where the generator produces high-quality data indistinguishable from the training distribution and the discriminator makes casual predictions. The obtained results seems to be this ideal case but probably this is only a poor convergence. If we look at some generated images (Figure 25), this conclusions must be the most probable.

| | **Discriminator** | **Generator** |
|---|---|---|
| Loss | 1.36 | 0.72 |

Table 2: GAN results

## Dimensional reduction and data visualization

Unsupervised learning is concerned with discovering structure in unlabeled data. Data visualization methods are important for modeling as they can be used to identify correlated or redundant features along with irrelevant features from raw or processed data. Conceivably, being able to identify and capture such characteristics in a dataset can help in designing better predictive models. This rapidly becomes impractical for datasets involving a large number of measured featured (such as images). Thus, in practice, we often have to perform dimensional reduction, project or embed the data onto a lower dimensional space, which we refer to as the latent space. In the following are presented the results obtained with different methods with the same goal: reduce the dimension of the data space of the FasionMNIST dataset and provide the best visualization.

In Figure 26 and 27 are shown the results obtained using the two already presented models, the autonencoder and the variational autoencoder respectively. It emerges that the representations are not the best possible ones, indeed there are is not a clear distinction between the different class clusters. However, specially for the second plot, there is a structure which seems to organize the clusters along horizontal lines. A more clear distinction and data visualization is provided by the t-SNE technique in 29. This non-linear technique after constructing a non-linear embedding map each point to low-dimensional space in a way to preserve the local structure in the data. This seems to be

the best technique to visualize the FshionMNIST dataset, even if it works in a similar way to the two implemented models. It is also reported the data visualization using PCA (28).

### Fine-tuning results

The last results of this work are shown in Table 3, where it can be seen the test accuracy and loss for the four different models. The best one is the loaded from the homework one, and this is probably because of it is optimized for that specific task and the hyperparameters are found in order to achieve the best performances in the classification. On the other hand, the results of the fine-tuned model from the autoencoder is quite good. Indeed the most interesting point lies in the fact that this classification is performed starting from a model trained for another task and retrained with a simple modification and for only 30 epochs. Moreover, the training was really fast for this model. The performances of the two ResNet models are worse in terms of final loss and in terms of computational time, but despite this the results are good and it is important to notice that the ResNet model was trained with a completely different dataset and this probably has lead to a slower convergence. The grater number of parameters has instead increased the time for the training.

All the trends of the losses reported in Figures 30, 31 and 32, which confirm that the models do not

| **Model** | **Test accuracy (%)** | **Test loss** | **Training time (s)** |
|---|---|---|---|
| AutoEncoder | 85.7 | 0.410 | 252 |
| ResNet18 (pre-trained) | 85.9 | 0.482 | 371 |
| ResNet18 (random weights) | 81.1 | 0.602 | 674 |
| Homework 1 | 87.8 | 0.318 | - |

Table 3: Fine-tuning test results

overfit the data and that they have learnt in the correct way, even if the accuracy loss of the ResNet model with pre-trained weights has an increasing trend in the last epochs, maybe it is because this models is too complex for this specific task and with less layers the performances could improve.

**Confusion matrices:** thanks to the confusion matrices (Figures 33, 34, 35 and 36), we can see where the algorithm made the main common mistakes in the classification and what are the classes more difficult to distinguish from each other. In fact, the shirt is most difficult class to identify correctly, often confused for a t-shirt, but this is not strange because they are very similar and even for a human is not always easy to discriminate them. Another difficult distinction is between the coat and the pullover, indeed also these two samples are not easy to distinguish, but it is interesting to notice that if from one side the coat is not always clearly identified, the pullover is rare that it is confused with a coat. Probably the reason is that the model has found better the main features of the pullover but the characteristic property of the coat are not yet found. Another thing to notice is in the difference between the classification of the AE and the model of the first homework. The former has some classes that is able to distinguish very well and has other classes where it is not always sure of what it is. On the other hand, the second model has a more distributed accuracy, in some single classes is worse than the AE but in the overall classification makes less mistakes.

## References

[1] `https://arxiv.org/pdf/1511.06434.pdf`

[2] `https://arxiv.org/pdf/1803.08823.pdf`

## Appendix

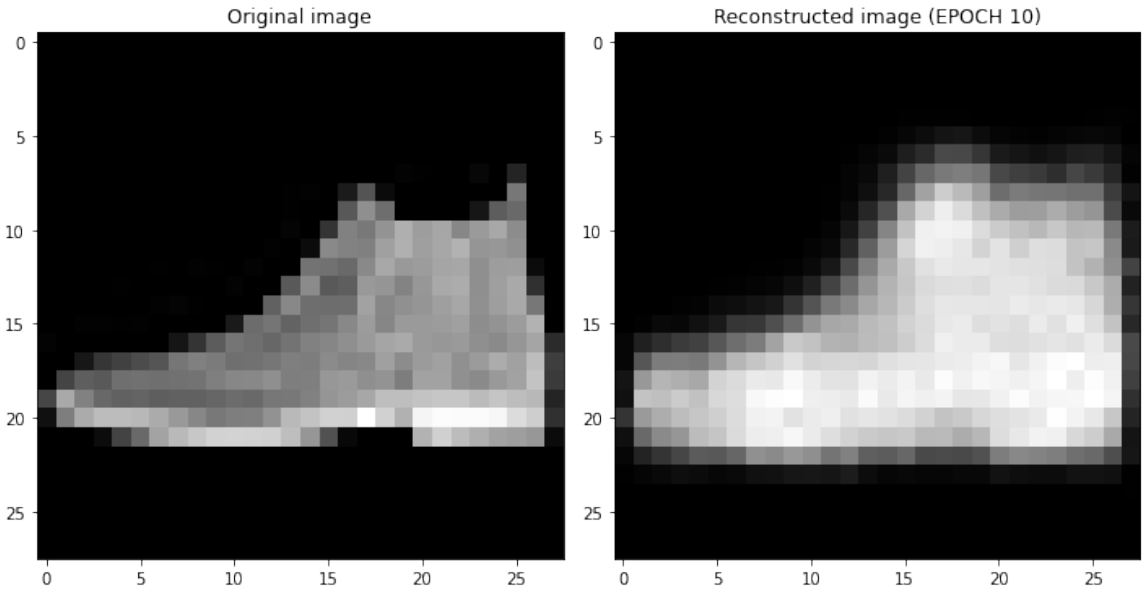


Figure 1: Autoencoder: original image and reconstructed image after first training

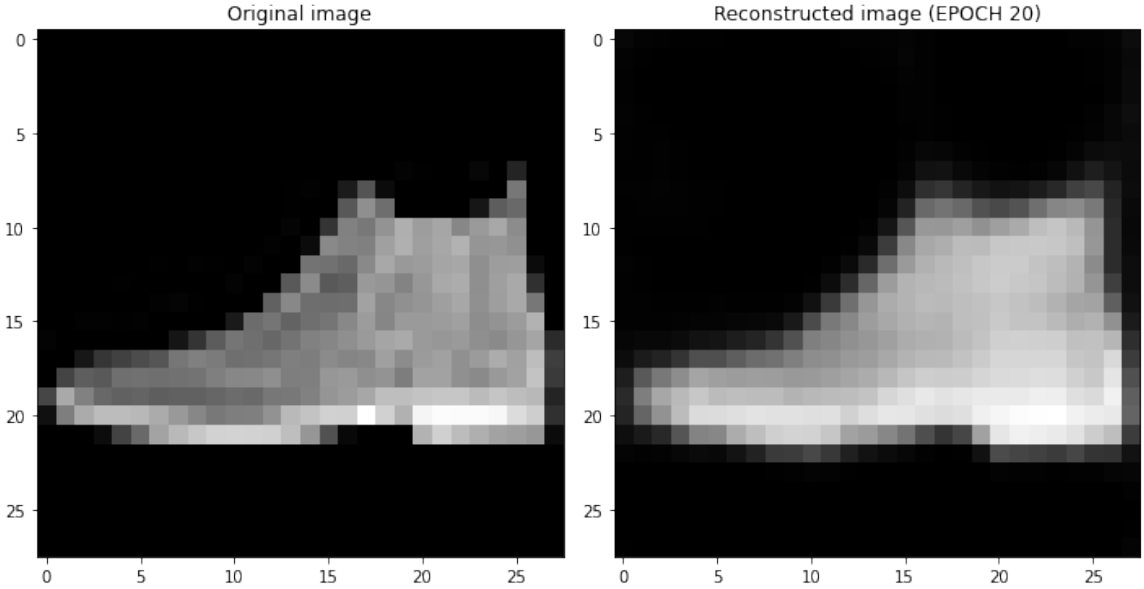


Figure 2: Autoencoder: original image and reconstructed image after k-fold training

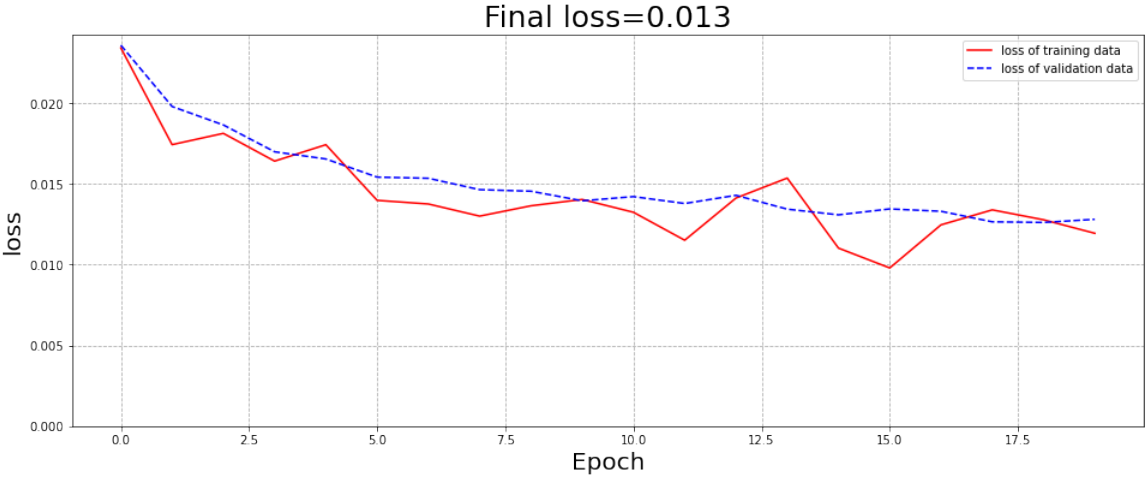


Figure 3: Autoencoder: loss of last fold k-fold cross validation

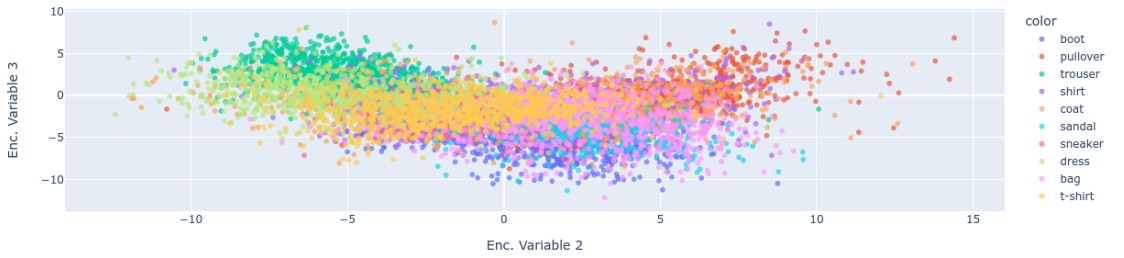


Figure 4: Autoencoder: representation of two latent space dimensions (2 and 3)

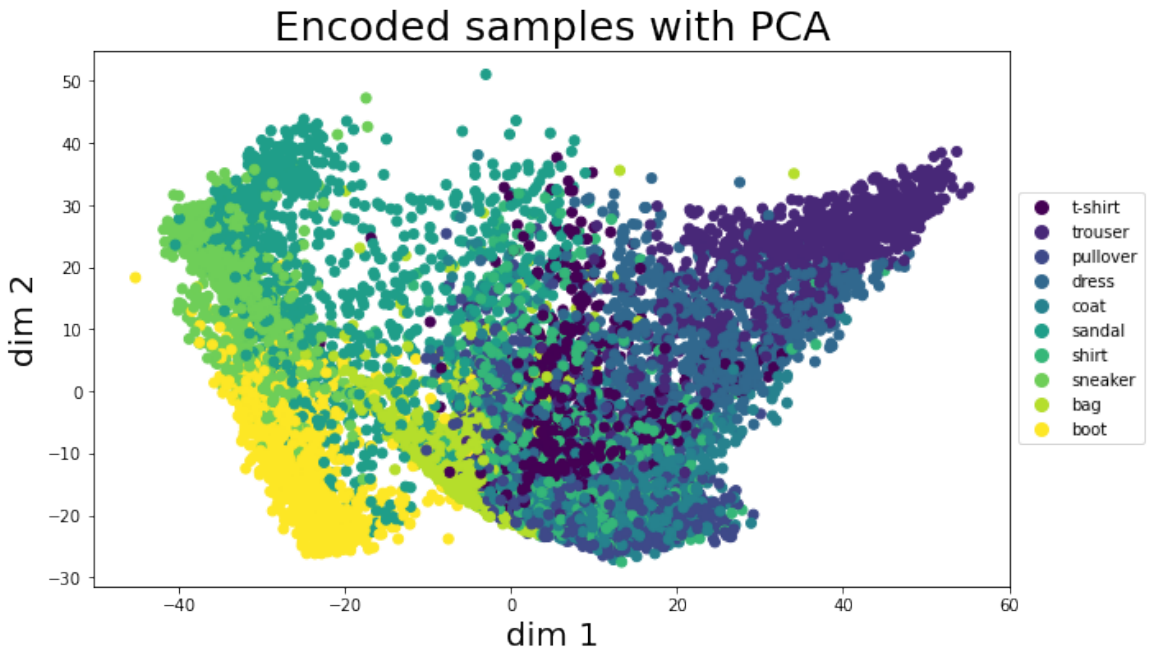


Figure 5: Autoencoder: representation of latent space with PCA

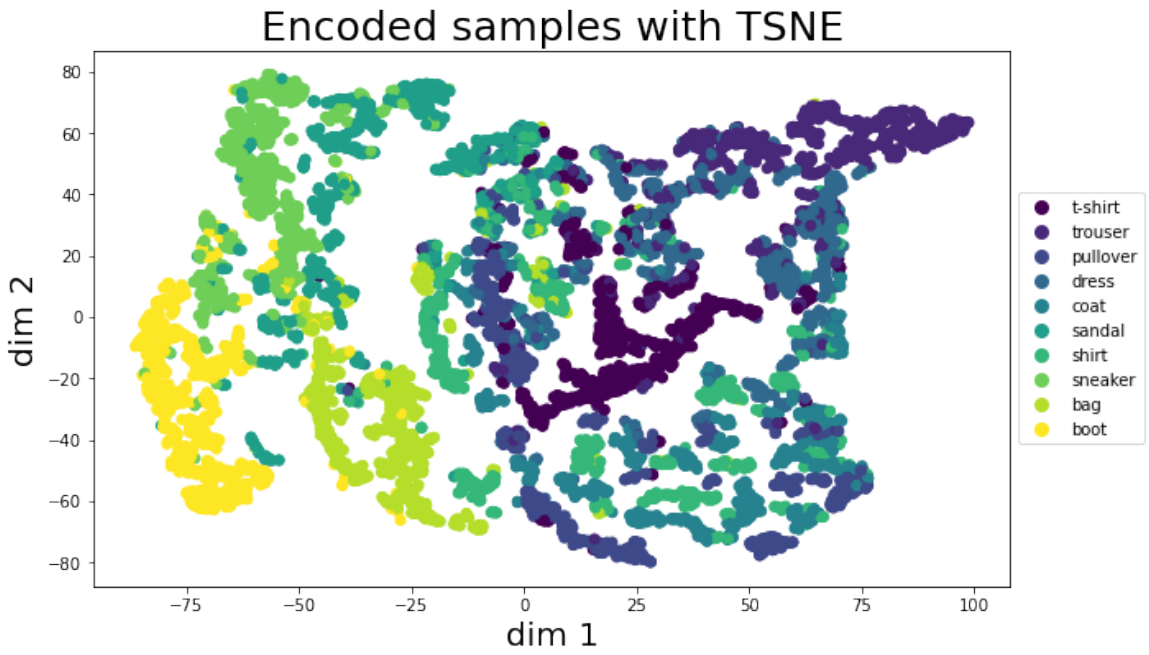


Figure 6: Autoencoder: representation of latent space with T-SNE

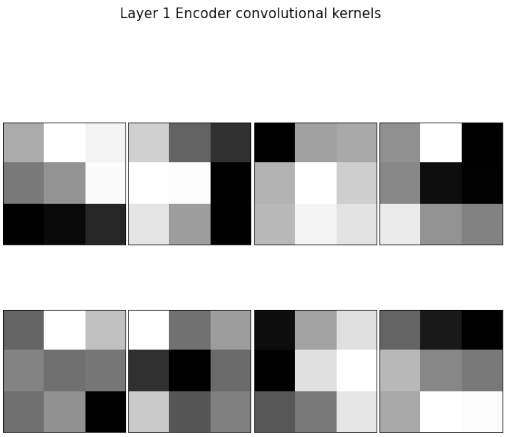


Figure 7: Autoencoder: filters of first encoder convolutional layer

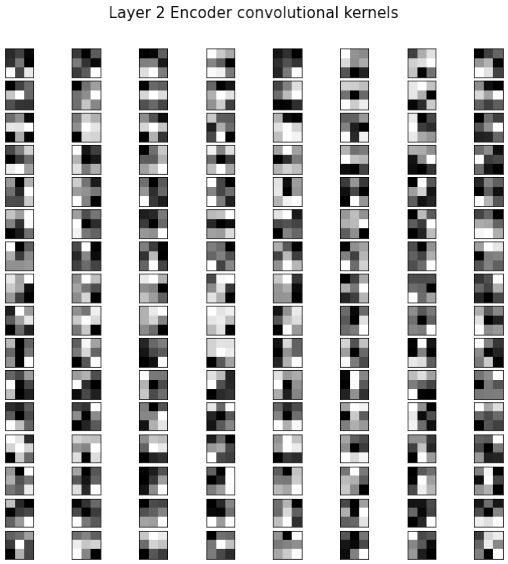


Figure 8: Autoencoder: filters of second encoder convolutional layer

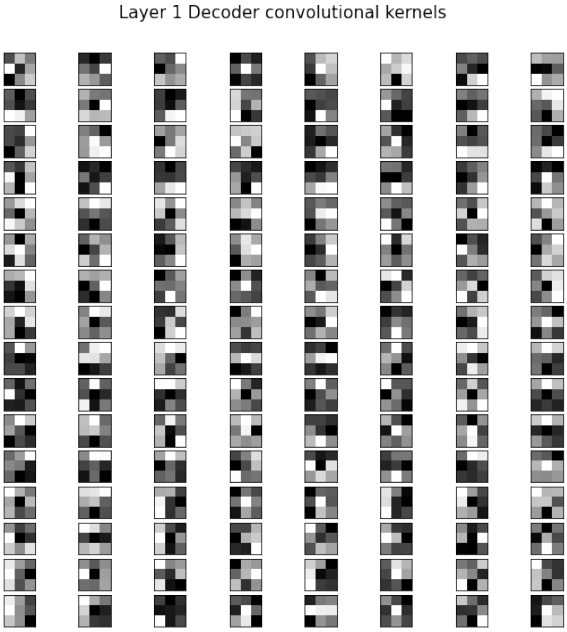


Figure 9: Autoencoder: filters of first decoder deconvolutional layer

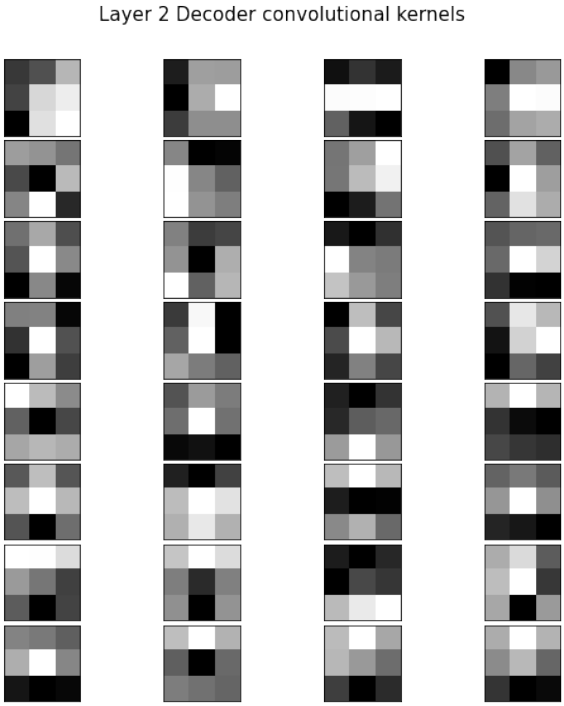


Figure 10: Autoencoder: filters of second decoder deconvolutional layer

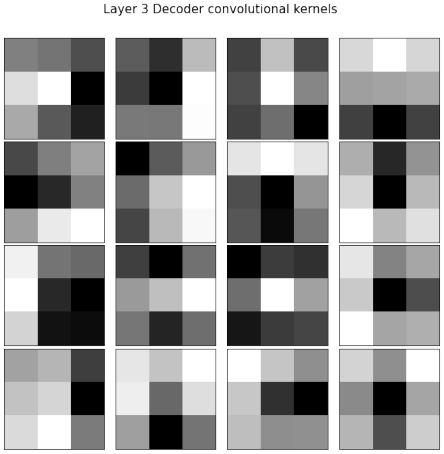


Figure 11: Autoencoder: filters of third decoder deconvolutional layer

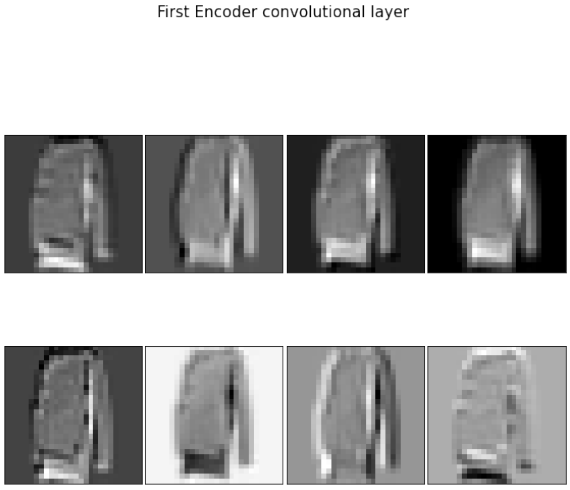


Figure 12: Autoencoder: activation of first encoder convolutional layer

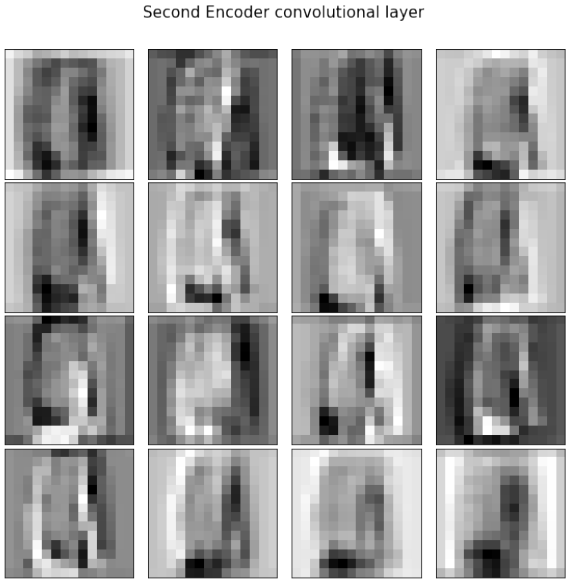


Figure 13: Autoencoder: activation of second encoder convolutional layer

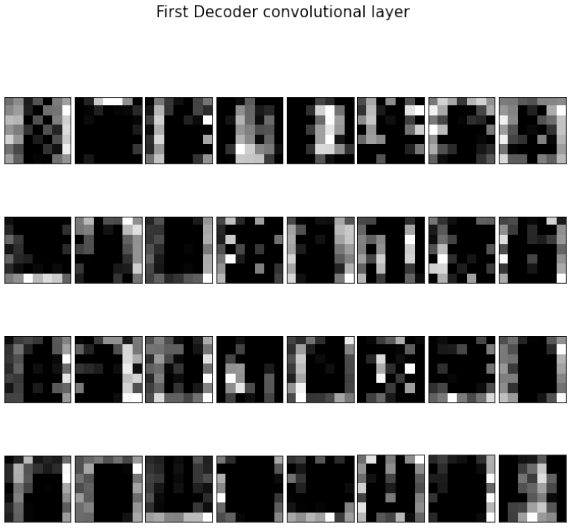


Figure 14: Autoencoder: activation of first decoder deconvolutional layer

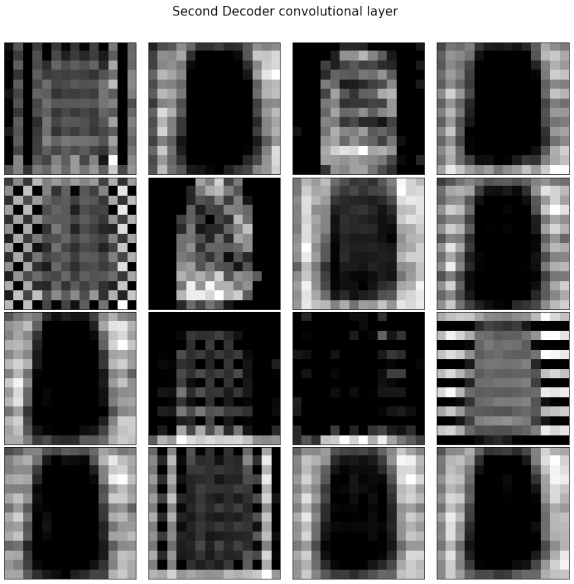


Figure 15: Autoencoder: activation of second decoder deconvolutional layer


Figure 16: VAE: reconstructed images

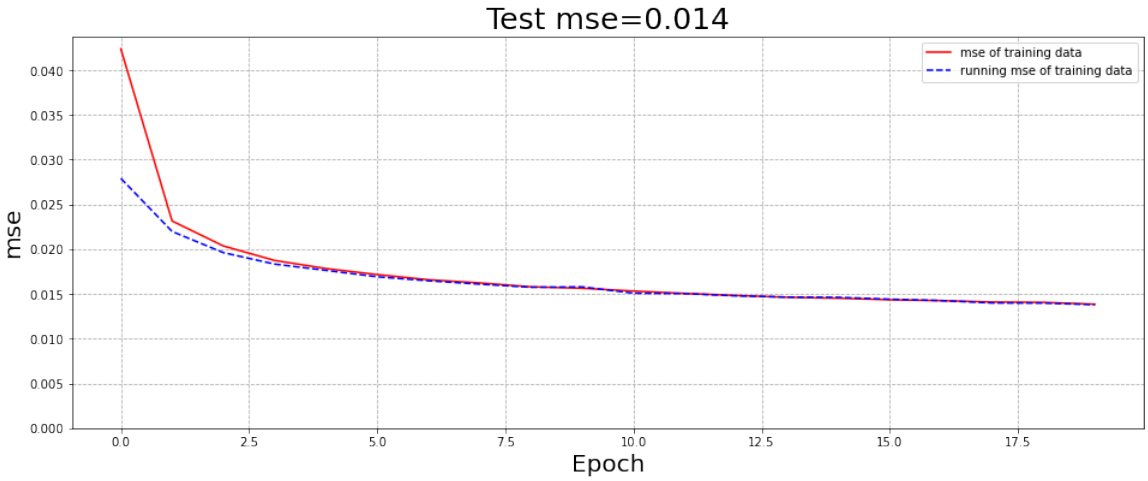


Figure 17: VAE: training and validation loss trend

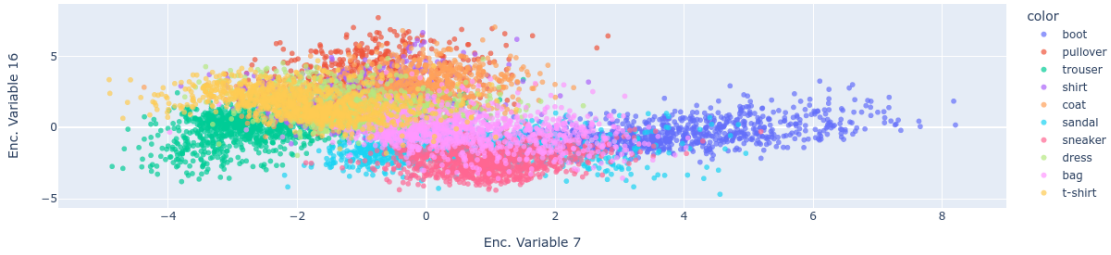


Figure 18: VAE: representation of two latent space dimensions (7 and 16)

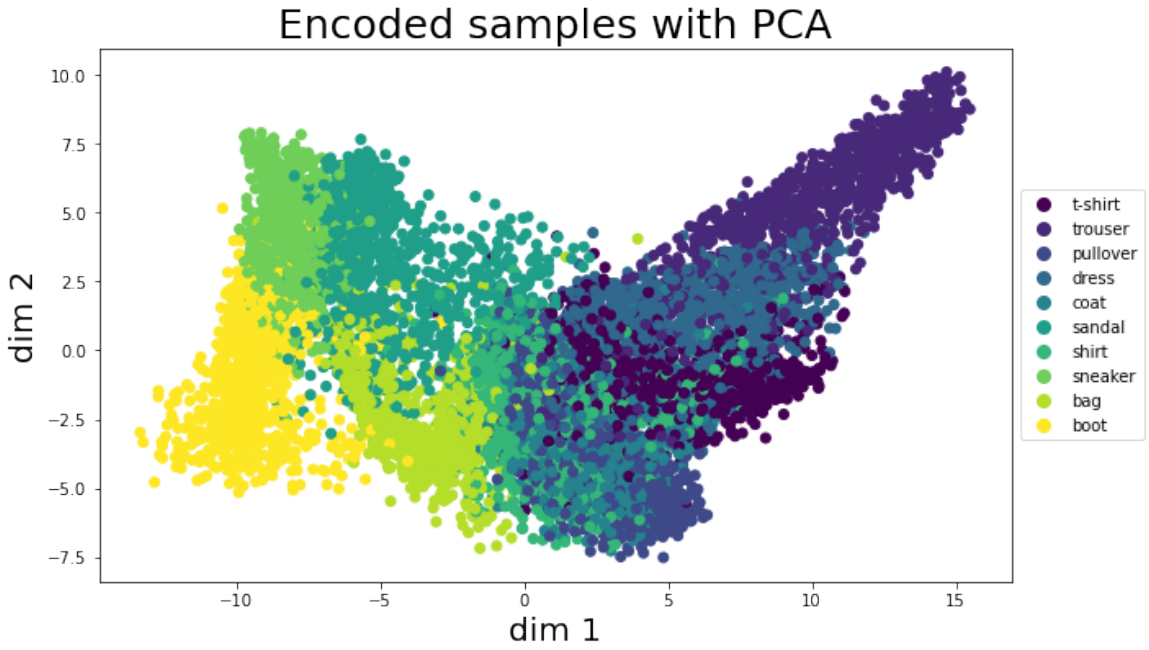


Figure 19: VAE: representation of latent space with PCA

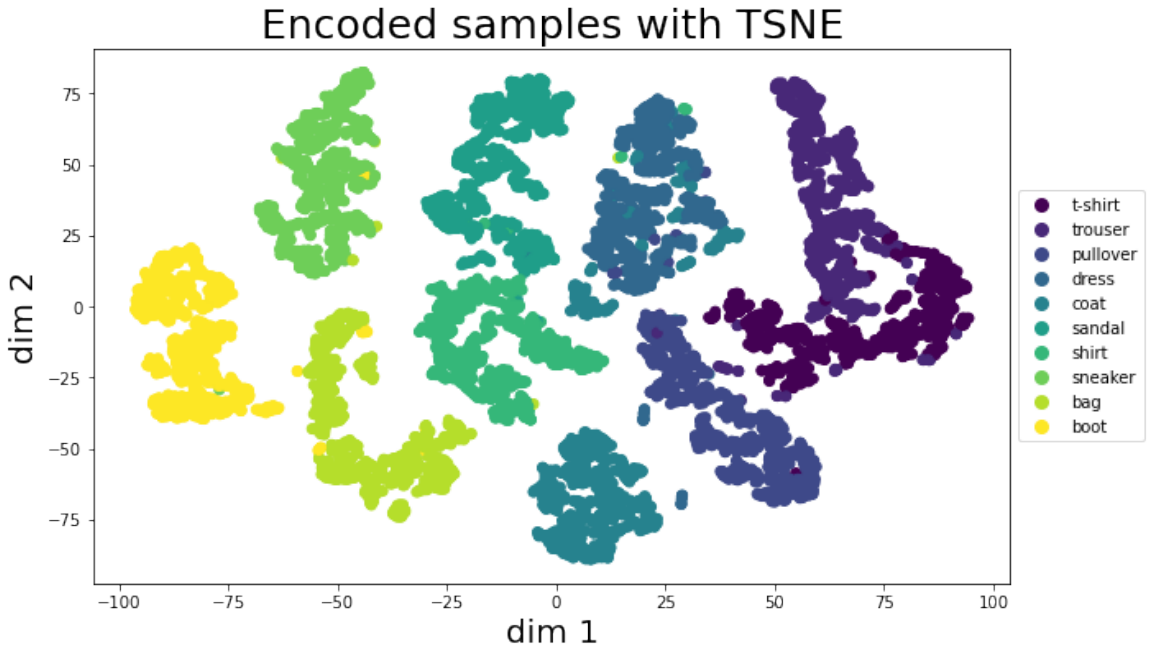


Figure 20: VAE: representation of latent space with T-SNE

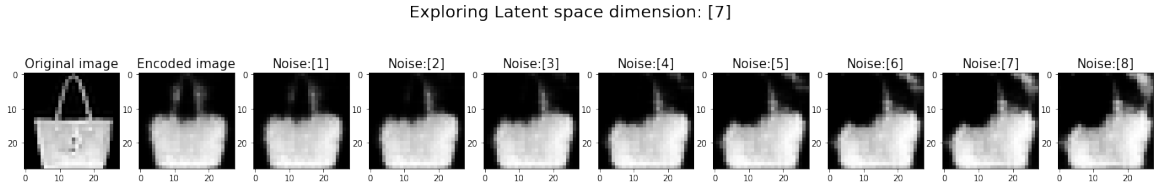


Figure 21: VAE: exploration of latent space dimensions 7

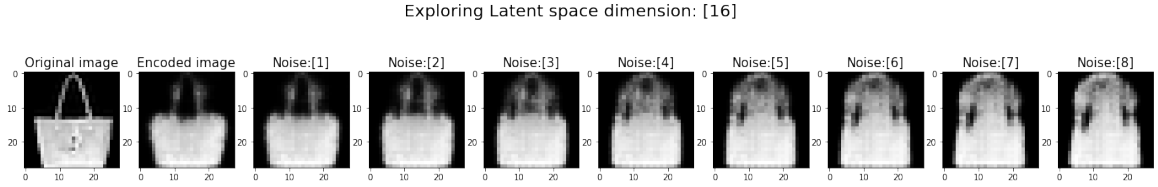


Figure 22: VAE: exploration of latent space dimensions 16

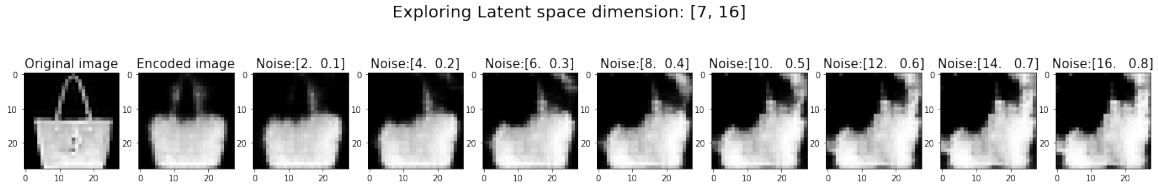


Figure 23: VAE: exploration of latent space dimensions 7 and 16


Figure 24: GAN: training and validation loss trend

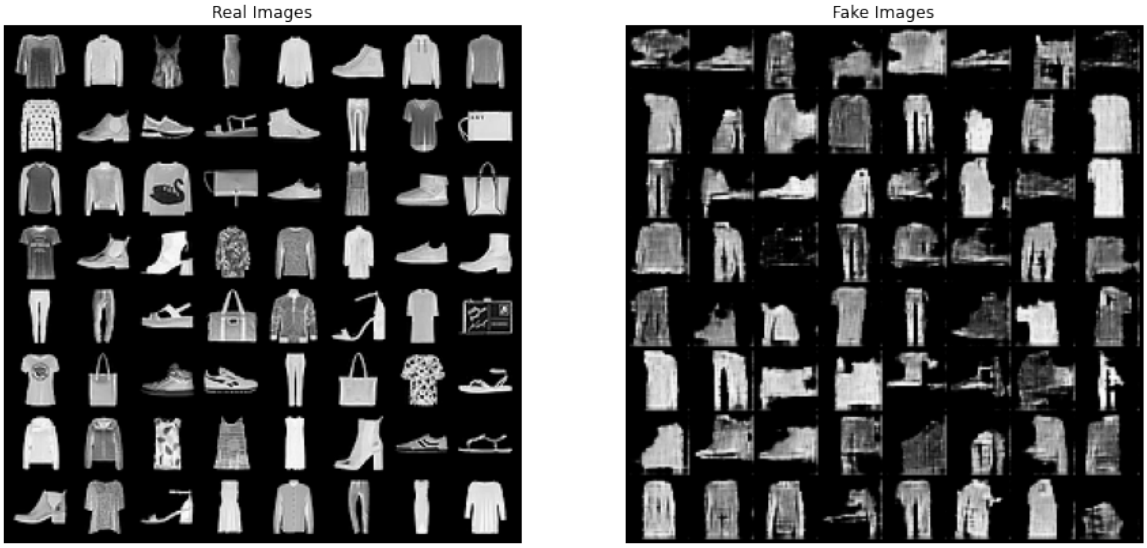


Figure 25: GAN: original and reconstructed images

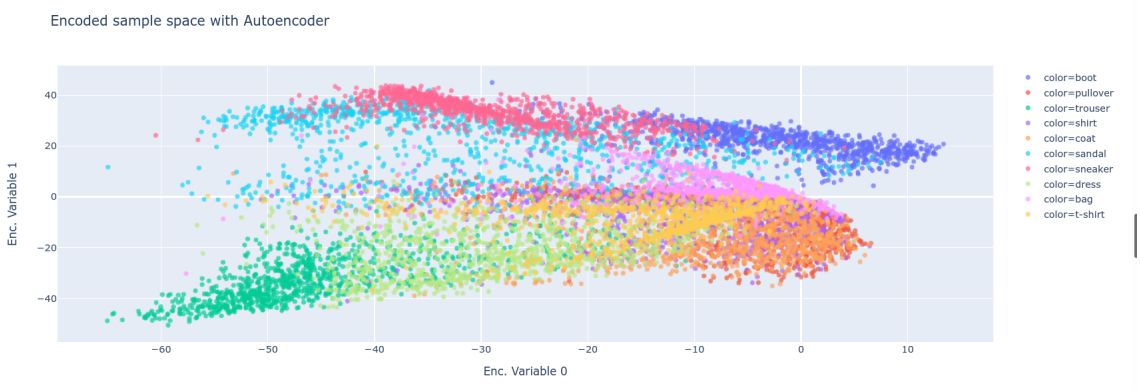


Figure 26: AE: representation of a latent space of dim 2

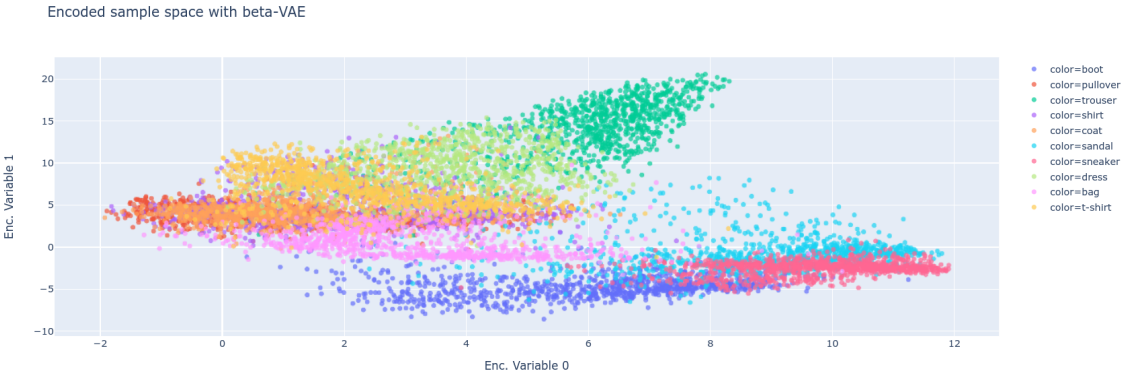


Figure 27: VAE: representation of a latent space of dim 2

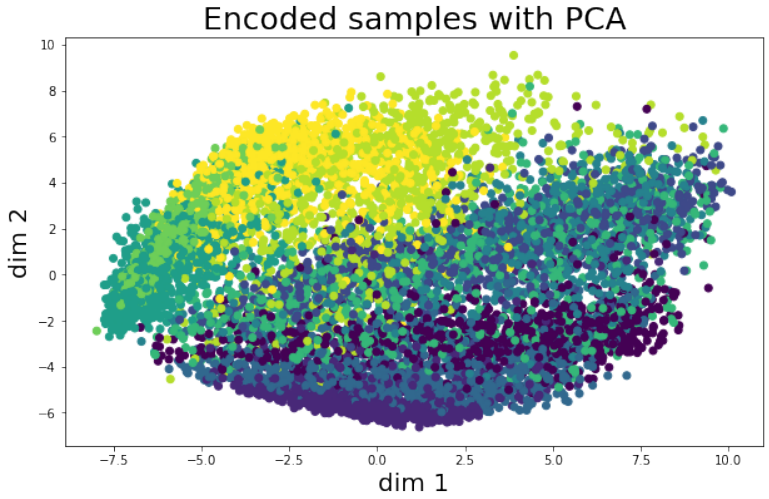


Figure 28: Data representation with PCA

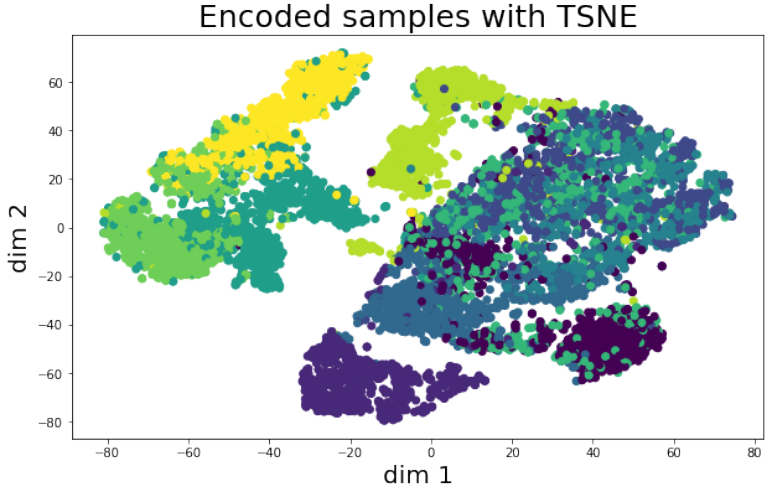




Figure 29: Data representation with PCA T-SNE

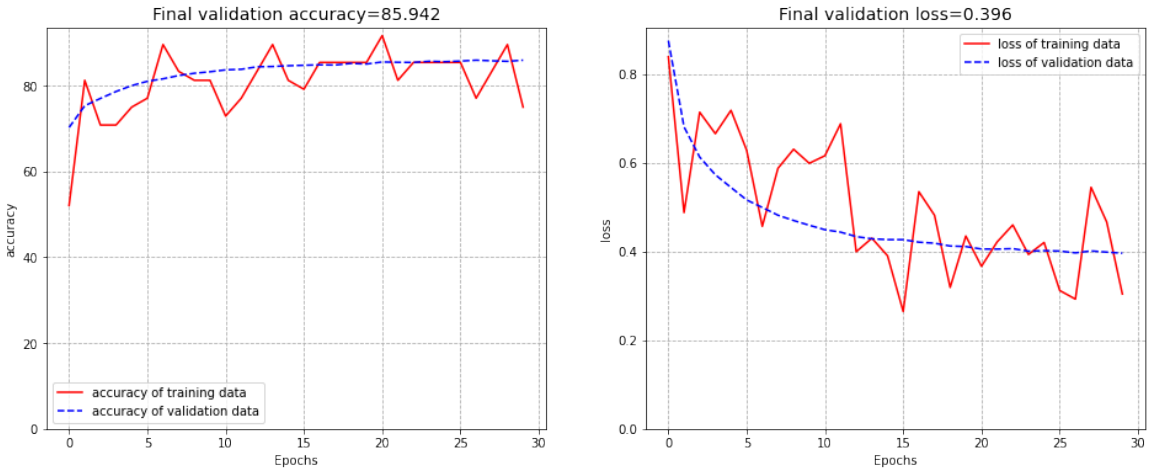


Figure 30: Fine tuning AE: training and validation loss trend

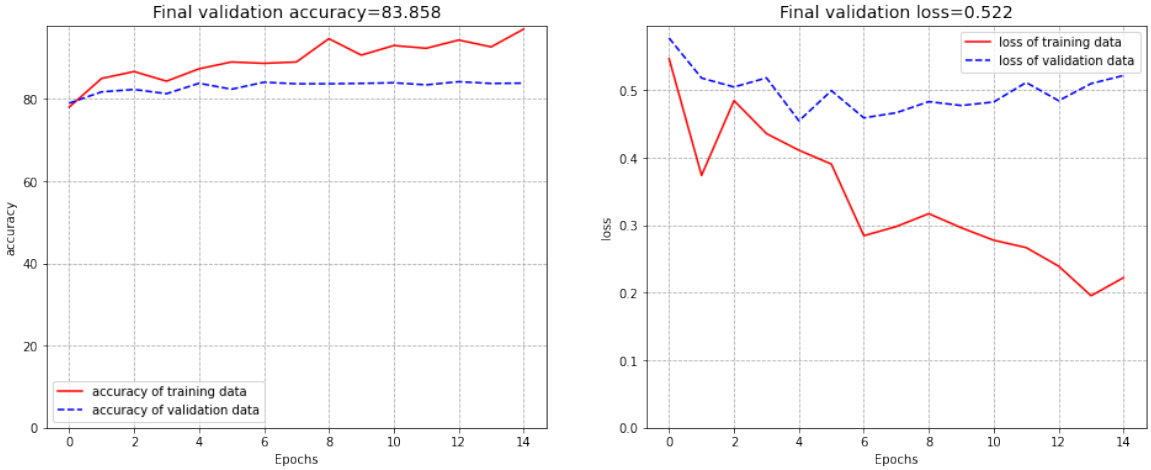


Figure 31: Fine tuning ResNet18 (pre trained model): training and validation loss trend

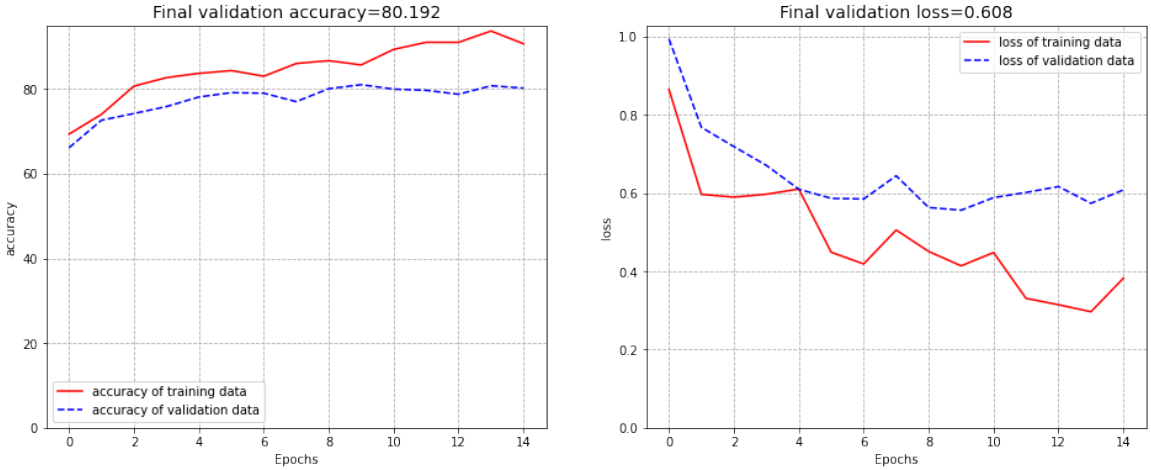


Figure 32: Fine tuning ResNet18 (random weights): training and validation loss trend

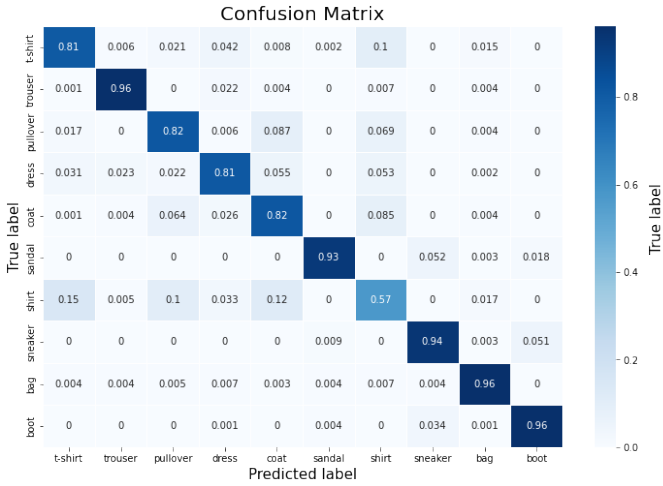


Figure 33: Fine tuning AE: confusion matrix

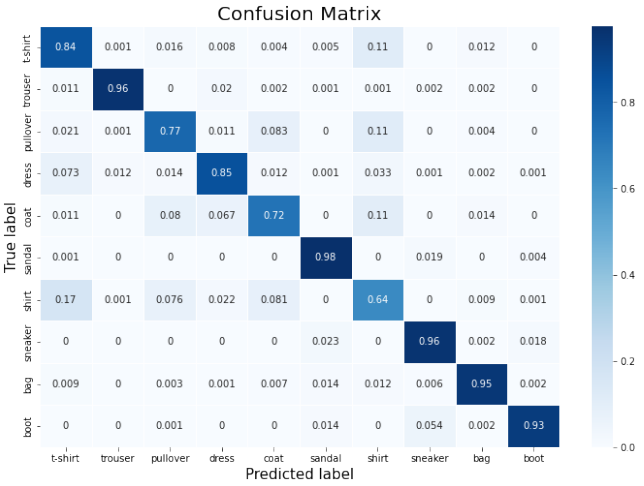


Figure 34: Fine tuning ResNet18 (pre trained model): confusion matrix

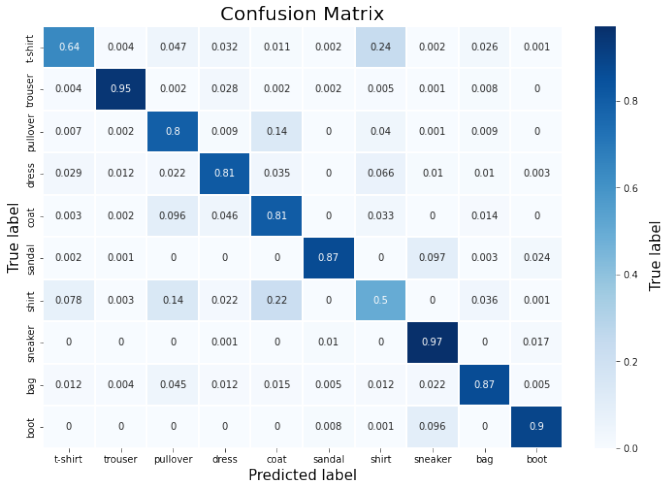


Figure 35: Fine tuning ResNet18 (random weights): confusion matrix

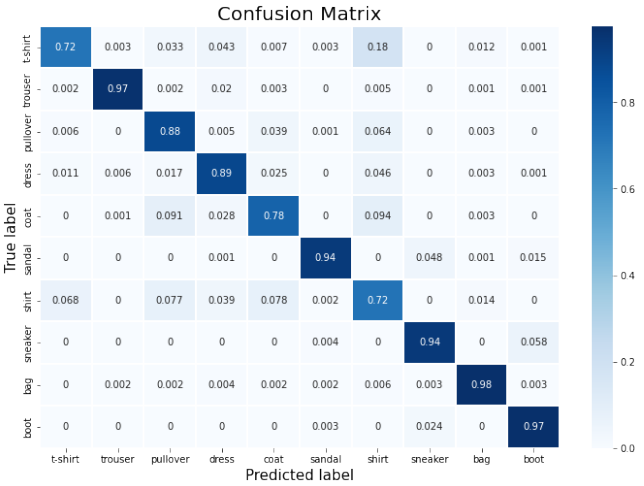


Figure 36: Homework 1: confusion matrix