CS 307 Final Project - Denoise Autoencoder

Junseok Yang

Abstract

An autoencoder is composed of two different networks: an encoder which accepts signals and generates a code whereas a decoder plays an opposite role, that is generating signals based on the code created from the encoder. Commonly, an encoder compresses high dimension input data and produces lower dimension codes, and a decoder recovers the original input data using the lower dimension codes. From this process, the main focus would be to build an autoencoder which can reconstruct data that is as identical as the original data. There are many ways to improve a model and yield better performances, and one method is to train an autoencoder using noisy data. By building a denoising autoencoder, it would be more robust in terms of overcoming hinders from noise and reconstructing original data that depict necessary information [2]. Two ultimate goals will be tested in this project: Check whether noisy data are helpful in terms of improving an autoencoder's performances, and find the amount of noise that yields the best performance scores.

Methodology

'FashionMNIST' dataset will be used in this project, and the first objective is to compare an autoencoder's performances with three different datasets: Dataset without any noise, another dataset with the Gaussian noise added (mean of 0 and standard deviation of 1), and finally dataset with Random Erasing. The second objective is to test the autoencoder with three different standard deviation values (0.01, 0.1, 0.5) of the Gaussian noise and compare which value yields the best performance.

The autoencoder used in this project is referred to as Anello's model. Both the encoder and decoder contain two different layers: convolution and linear layers. Input data first meets the encoder with three steps that would split into smaller pieces. After flattening the pieces, a linear layer eventually lowers the dimension of the data and sends them to a latent space. The decoder goes exactly the opposite direction. The data in the latent space enter the linear layer of the decoder first, increasing the dimension back. Followed by an unflattening process, a transposed convolution layer of the decoder finally gathers the data pieces into a new image or reconstructed image [1].

Objective 1

In the first section, the main focus is to compare autoencoders' performances trained with three different datasets (Normal, Gaussian noise, RandomErasing) and check whether the initial assumption, which is a denoise autoencoder is more robust and shows improved performances, is correct or not.

Data Images Comparison 1

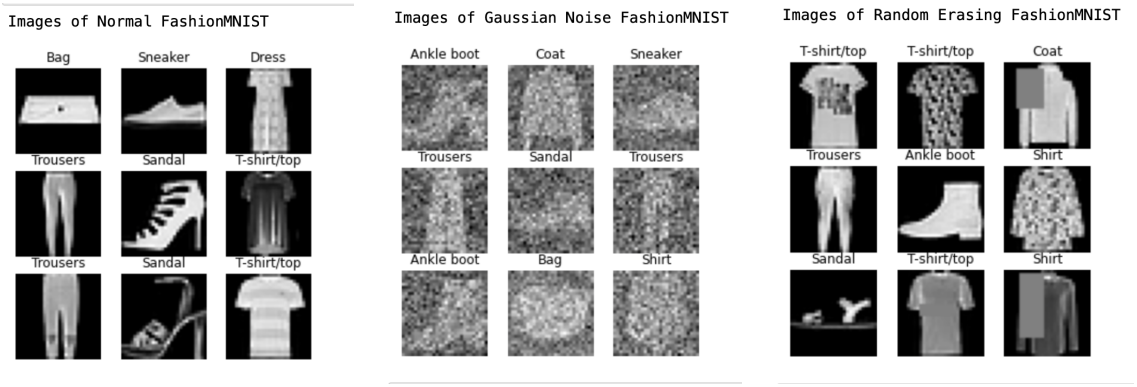

Figure 1: Images of three different FashionMNIST datasets

From the visualizations above, there are clear distinctions between three different datasets. The images of the Gaussian noise dataset are blurred and thus only able to detect the overall shape of fashion items. Any details such as logos are almost erased. On the other hand, the RandomErasing dataset displays identical images to the original dataset except that there are gray boxes in the images that hide some proportion of fashion items. For instance, some boxes may blind significant details like a string part in the sandal while the box in the T-shirt / top is inside the item, still able to figure out its label.

Autoencoder Evaluation 1 - Average Loss

Dataset \ Average Loss	**Epoch 1 (Train / Test)**	**Epoch 20 (Train / Test)**
Normal	0.848 / 0.675	0.601 / 0.599
Gaussian Noise	1.860 / 1.689	1.600 / 1.598
RandomErasing	0.922 / 0.679	0.569 / 0.567

Table 1: Train and test average loss of three different datasets

The three different autoencoders were trained and tested up to 20 epochs, and their loss graphs show a similar trend; both the train and test losses dramatically drop around the first three epochs and no significant drops are detected afterwards (Appendix figure 2, 4, and 6). The autoencoder trained with the RandomErasing dataset yielded the lowest train and test average losses. For the normal autoencoder, it has slightly higher average losses than the RandomErasing, but the reconstructed images from both autoencoders do not show significant differences (Appendix figure 1 and 5). On the other hand, the Gaussian autoencoder performed relatively worse than these two autoencoders. Not only are the average losses larger, its reconstructed images still contain some blurrinesses compared to other two autoencoders' recovered images (Appendix figure 3).

Autoencoder Evaluation 1 - Visualizations of Latent Spaces

Another method to differentiate these autoencoders' performances is to visualize their encoded samples or latent spaces with t-SNE [1]. t-SNE is one of dimensionality reduction algorithms which converts high dimension data into a lower dimension and visualizes observations with a scatter plot that is relatively easy to see and interpret . "The idea behind t-SNE is to find a two-dimensional representation of the data that preserves the distances between points as best as possible". The main point of the idea above and benefit of using t-SNE is

that although the algorithm does not necessarily display the exact distances between observations, it does preserve clustering structures. Comparing the structures with labels, it is possible to figure which clusters are well-cohesive with their own labels and separated to other clusters [3]. With the perplexity value of 30 and random state of 307, belows are the t-SNE plots of the encoded images of three different test datasets.,

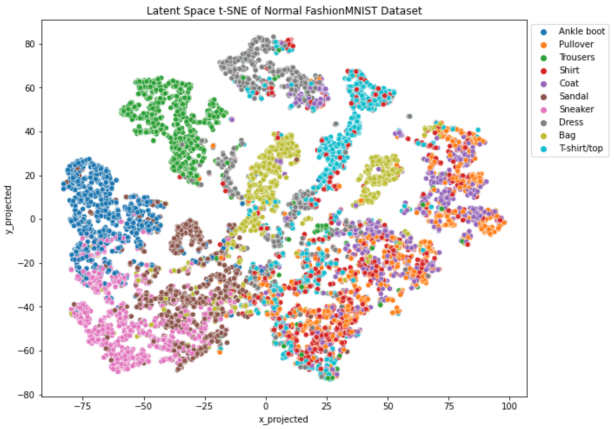

Figure 2: t-SNE of encoded Normal FashionMNIST samples

Although some fashion items that have similar shapes such as ‘Shirt’ (Red) and ‘Pullover’ (Orange) are not concentrated but rather widely spread out to other items clusters, items with unique traits like ‘Trousers’ (Green) form their own clusters with less other items’ observations. Overall, the clusters of the normal dataset’s t-SNE plot are mostly well separated.

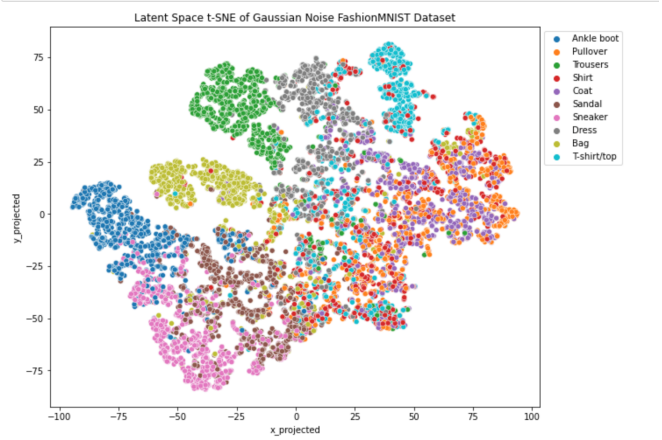

Figure 3: t-SNE of encoded Gaussian noise FashionMNIST samples

Contrary to the normal dataset’s t-SNE plot, the Gaussian noise dataset’s clusters are somewhat closer to each other and have more overlaps, specifically difficult to see clear boundaries between ‘Sneakers’ (pink), ‘Sandal’ (brown), ‘T-shirt/top’ (skyblue), and mixed of ‘Coat’ (purple), ‘Pullover’ (orange) and ‘Shirt’ (red). This may be due to the blurriness of the Gaussian noise remaining in the encoded images, difficult to distinguish and only able to detect the overall shape of the items.

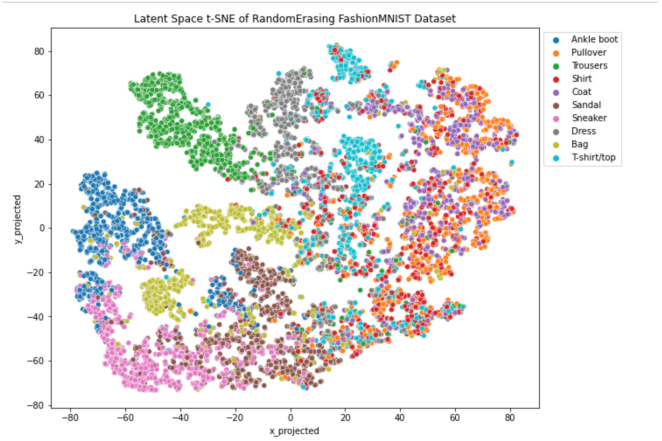

Figure 4: t-SNE of encoded RandomErasing FashionMNIST samples

Lastly, the RandomErasing dataset's t-SNE plot shows a similar structure to the normal dataset. Some unique fashion items are well-cohesive and separated, and even more white spaces can be found between clusters compared to the normal t-SNE. Considering that the RandomErasing autoencoder yielded the lowest average train and test losses, this result was reflected in the visualization.

Conclusion 1

Comparing two different criteria of the autoencoders' performances, it turns out that the RandomErasing autoencoder performed the best. Although there was not a big difference between the RandomErasing and normal autoencoders, it is still obvious that the former's performances were better. One interesting point is that the Gaussian noise autoencoder performed significantly worse compared to these two autoencoders. The expectation made in the beginning was not correct throughout the procedure so far.

Objective 2

Starting from this section, the main procedure shifts from testing different types of noise to different amounts of noise. Considering that the performances of the Gaussian noise (σ=1) autoencoder were significantly poor, the major aim would be to train autoencoders with three different standard deviations of the Gaussian noise (σ=0.01 / σ=0.1 / σ=0.5) and figure out the best amount of the noise and whether the noise do help improve the autoencoder's overall performances and yield better results than the autoencoder trained with a plain data.

Data Images Comparison 2

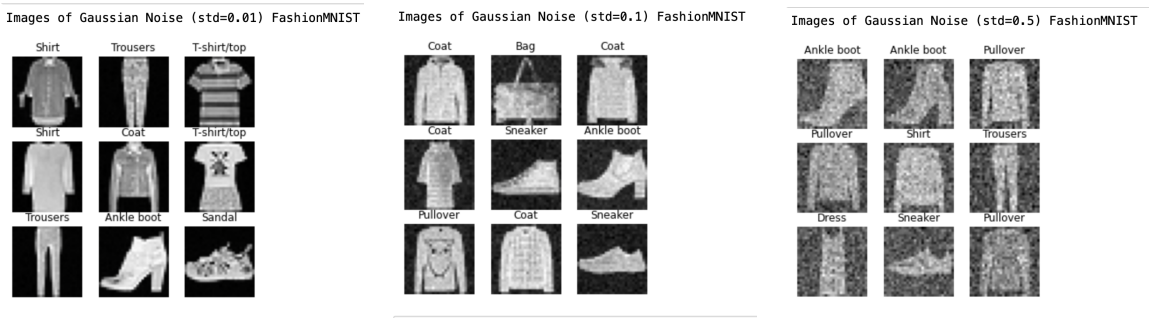

Figure 5: Images of different Gaussian noise datasets (σ=0.01 / σ=0.1 / σ=0.5)

From figure 6, it is almost impossible to find significant differences between the images of the first Gaussian noise (σ=0.01) and the normal dataset. For the second Gaussian noise (σ=0.1) images, although the images are blurrier than the first, some crucial details are still detectable. Lastly, the third Gaussian noise (σ=0.5) images are somewhat similar to the Gaussian noise (σ=1) images, fairly fuzzy and only able to see the overall shape.

Autoencoder Evaluation 2 - Average Loss

Dataset \ Average Loss	**Epoch 1 (Train / Test)**	**Epoch 20 (Train / Test)**
Gaussian σ=0.01	0.949 / 0.731	0.600 / 0.598
Gaussian σ=0.1	1.014 / 0.734	0.612 / 0.610
Gaussian σ=0.5	1.103 / 0.929	0.852 / 0.850

Table 2: Train and test average loss of three different Gaussian noise datasets

The loss graphs of three different standard deviations show an analogous trend to the graphs in the first autoencoder evaluation section (Appendix figure 8, 10, and 12). There seems to be huge drops in the beginning for both the train and test losses, and remain fairly the same average losses throughout the rest of epochs. From the table above, the differences of train and test average loss between three different standard deviations are somewhat significant. It is surprising that the autoencoder trained with the Gaussian noise of σ=0.01 has slightly lower average losses than the normal autoencoder (0.601 / 0.599). The other two autoencoders performed worse than the normal encoder.

Autoencoder Evaluation 2 - Visualizations of Latent Spaces

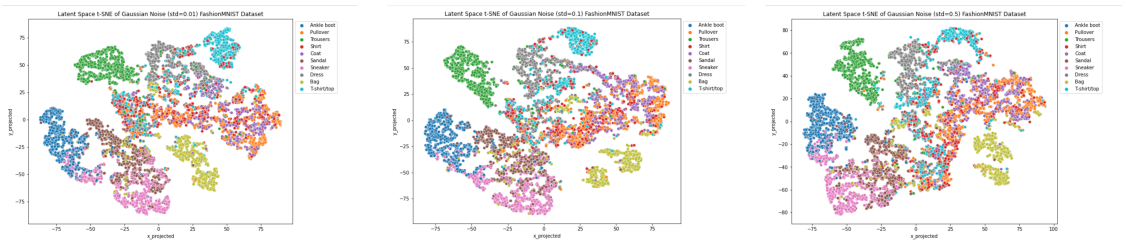

Figure 6: t-SNE of three different Gaussian noise (σ=0.01 / σ=0.1 / σ=0.5) encoded samples

Similar analysis can be done in this part as well. Overall, the t-SNE plots contain more well-cohesive and separated clusters compared to the t-SNE of the Gaussian noise with σ=1 (Figure 3). Thinking that the average losses differences were somewhat notable between different standard deviations, it is surprising that there is no relatively significant dissimilarity found in the t-SNE above.

Conclusion 2

The overall average losses of three different standard deviations were comparatively lower than the Gaussian noise with σ=1. Among them, the autoencoder trained with the Gaussian noise with σ=0.01 indeed yielded lower losses than the normal autoencoder. Beside the loss comparison, there was no remarkable variation in the t-SNE plots in figure 6.

Conclusion / Reflection / Future Plan

Throughout two different procedures, it is clearly shown that the RandomErasing autoencoder had the best performances overall. Although the autoencoder trained with the Gaussian noise of σ=1 showed relatively high average losses and poor clustering structure in the first procedure, the data of the Gaussian noise with σ=0.01 improved its autoencoder to have lower average losses and more cohesive and separated clustering structure than the normal autoencoder in the second procedure. To sum up, the project was able to figure out the general assumption about a denoise autoencoder to be correct, which was that noisy datasets (either the RandomErasing or with a small amount of the Gaussian noise in this project) aid an autoencoder to be more robust and yield better performances. In the future, it may be a good idea to figure out a more robust autoencoder by building a new structure of the autoencoder and test with more different types and amounts of noise.

Overall, although it was not an easy project, it was a great time to practice and develop skills of planning and doing diverse analyses in the project by myself. There are still many things I need to hone and improve, and I am embarrassed and even disappointed in myself that I was not able to build my own autoencoder and other relevant codes in the project, but instead refer to someone’s codes due to lack of both the programming skill and sufficient understanding of neural networks. Therefore, I am planning to spend more time studying and reviewing important concepts and applications again to strengthen the fundamental knowledge of neural networks and deep learning.

Appendix

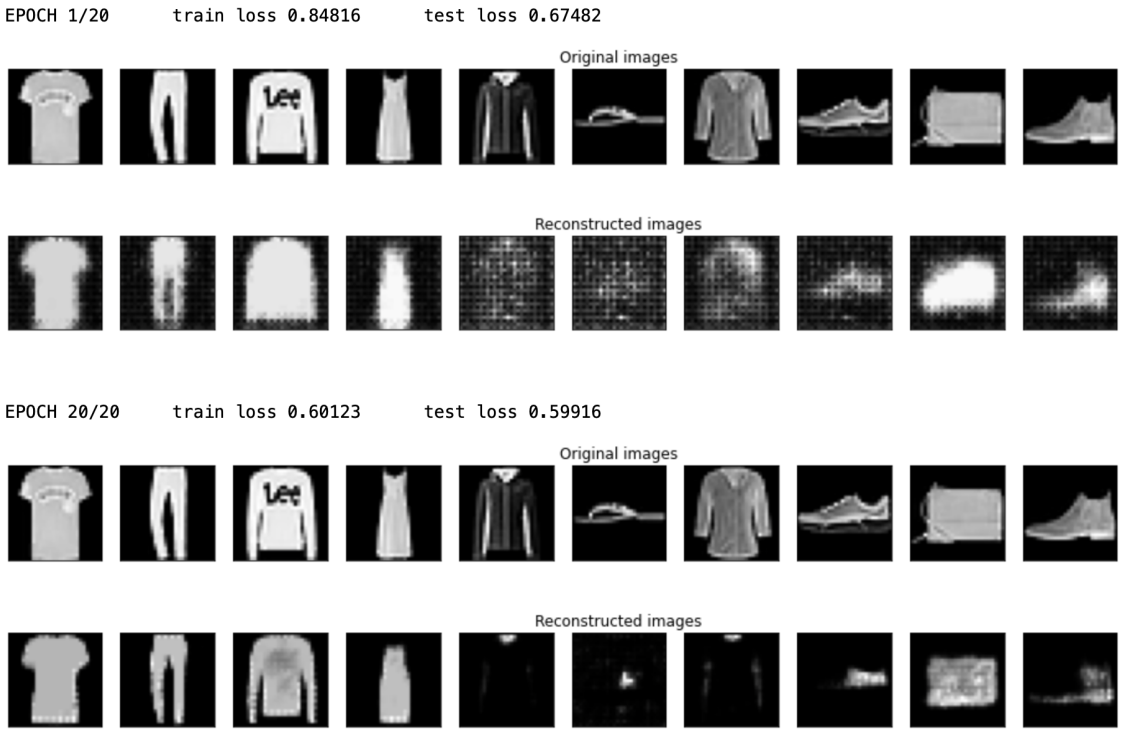

Figure 1: Train / test losses and original / reconstructed images of first and last (20) epochs of the normal dataset

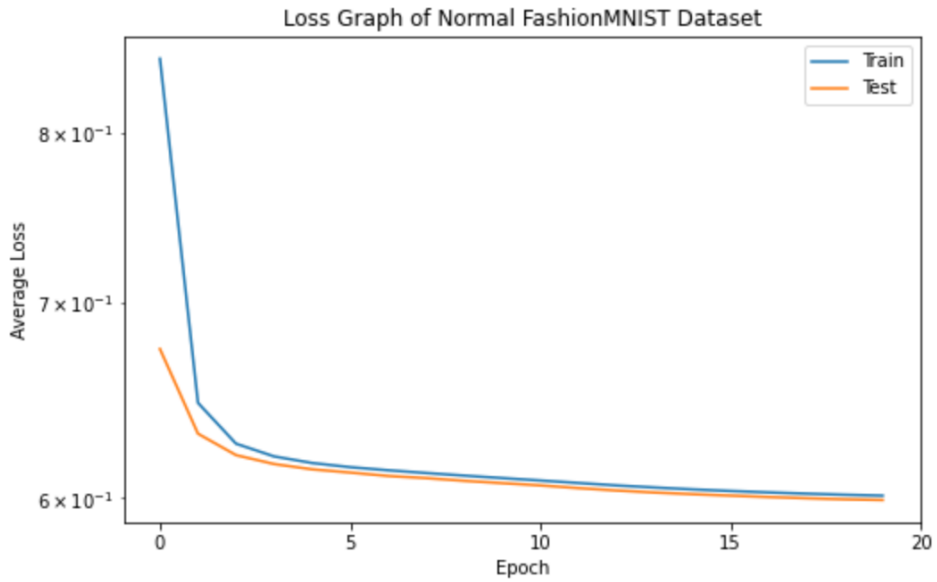

Figure 2: Loss graph of the normal dataset

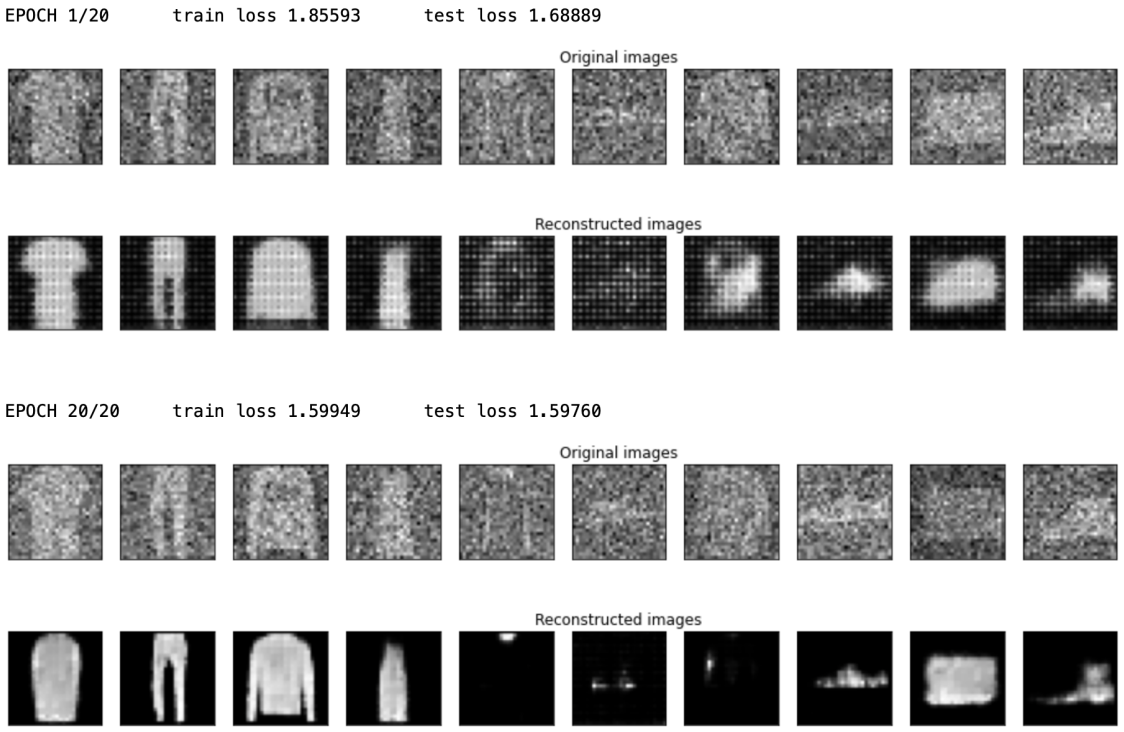

Figure 3: Train / test losses and original / reconstructed images of first and last (20) epochs of the Gaussian noise dataset (μ=0, σ=0.01)

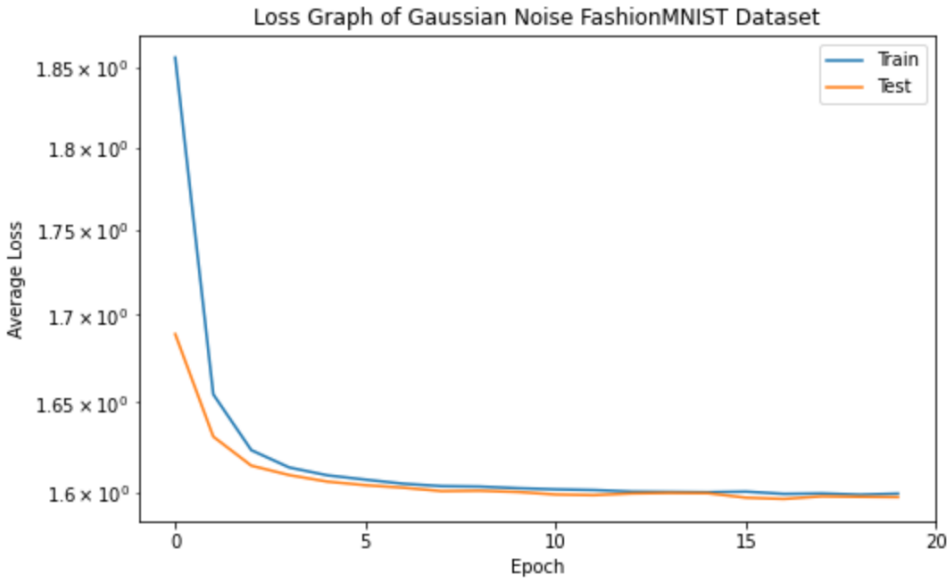

Figure 4: Loss graph of the Gaussian noise dataset (μ=0, σ=1)

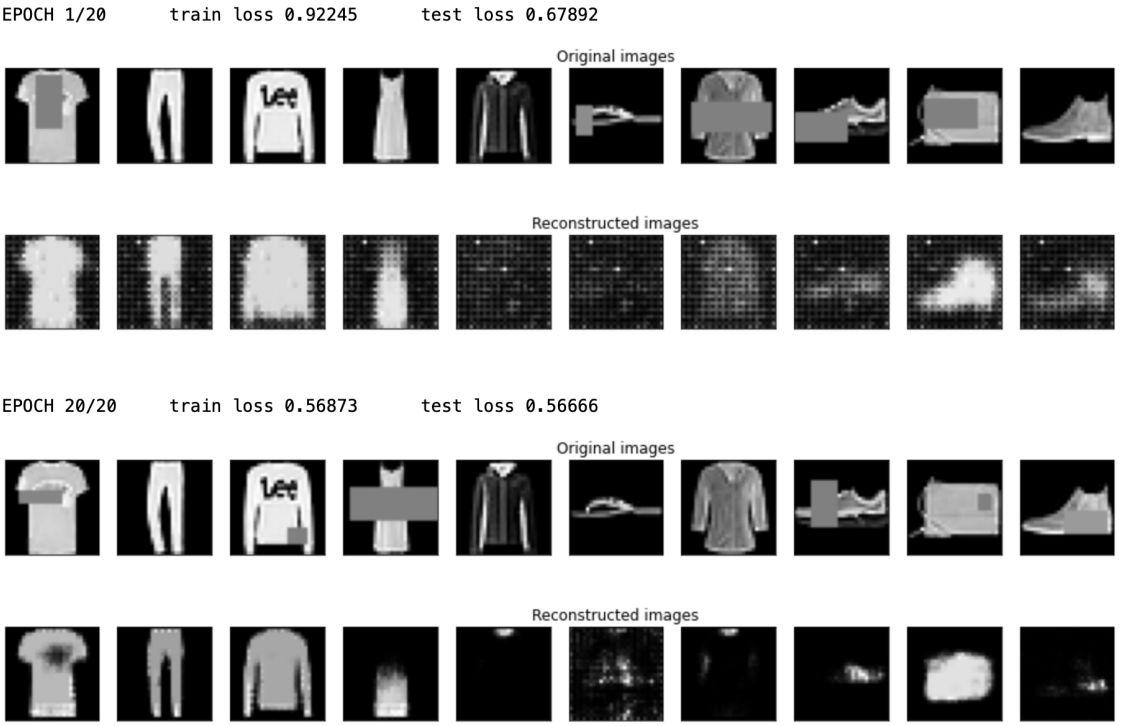

Figure 5: Train / test losses and original / reconstructed images of first and last (20) epochs of the RandomErasing dataset

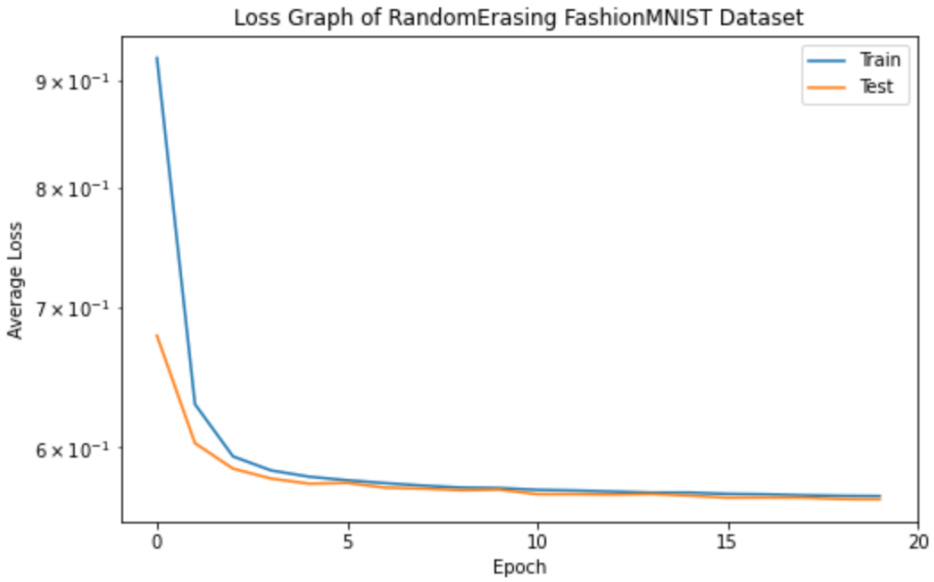

Figure 6: Loss graph of the RandomErasing dataset

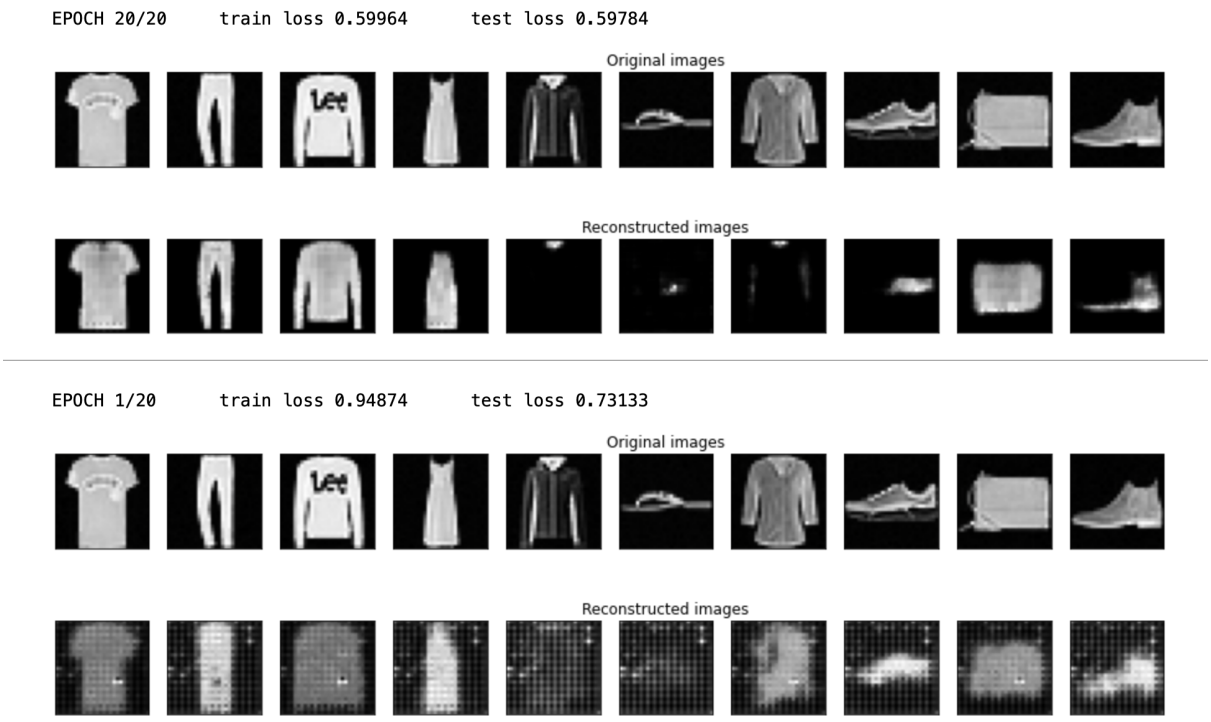

Figure 7: Train / test losses and original / reconstructed images of first and last (20) epochs of the Gaussian noise dataset (μ=0, σ=0.01)

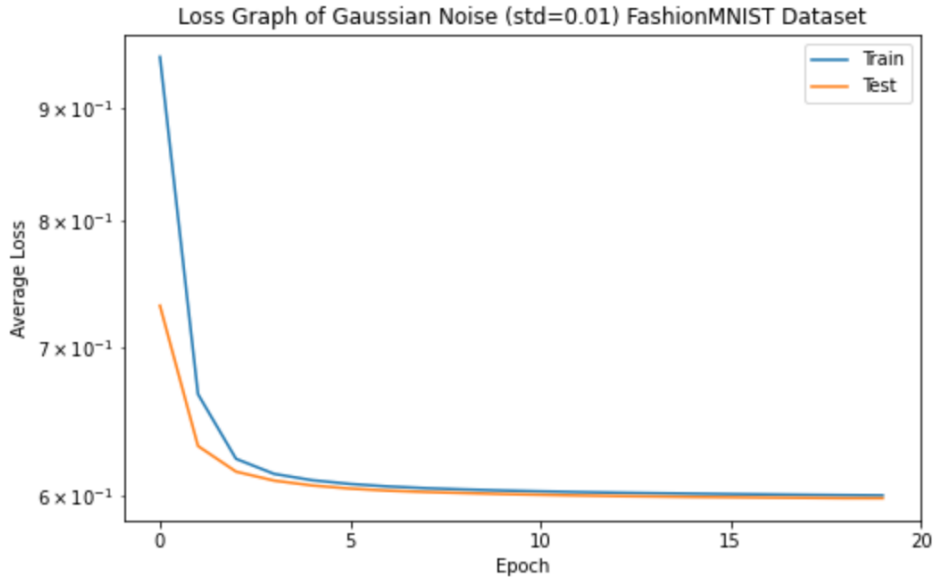

Figure 8: Loss graph of the Gaussian noise dataset (μ=0, σ=0.01)

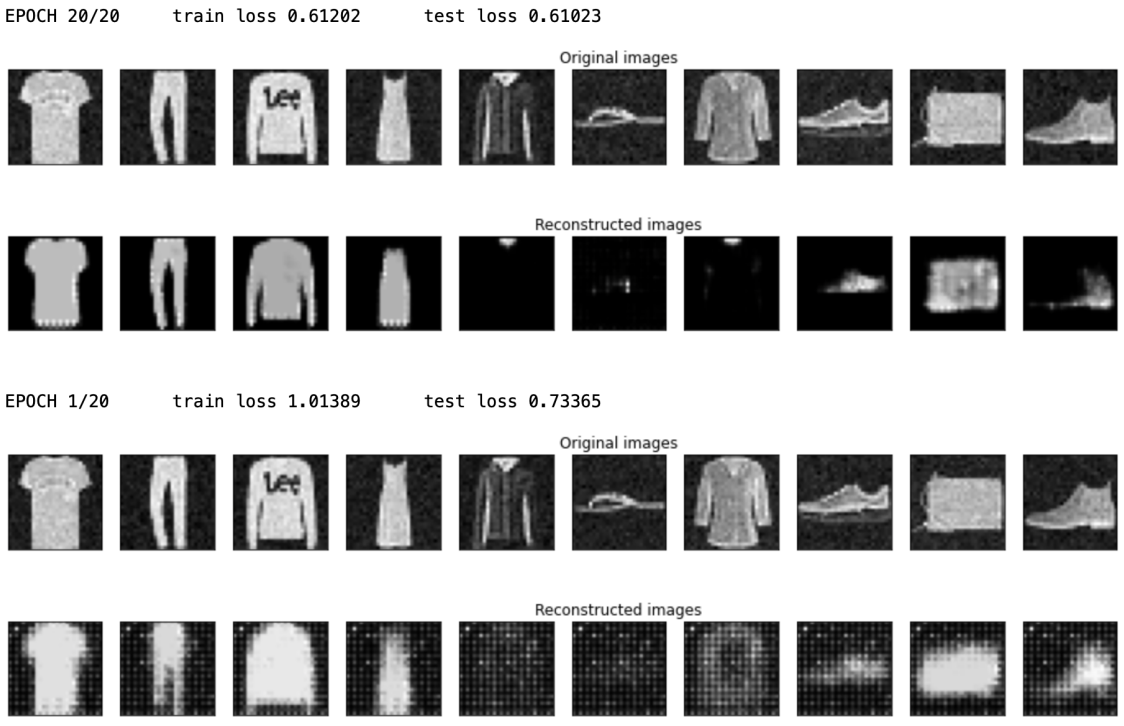

Figure 9: Train / test losses and original / reconstructed images of first and last (20) epochs of the Gaussian noise dataset (μ=0, σ=0.1)

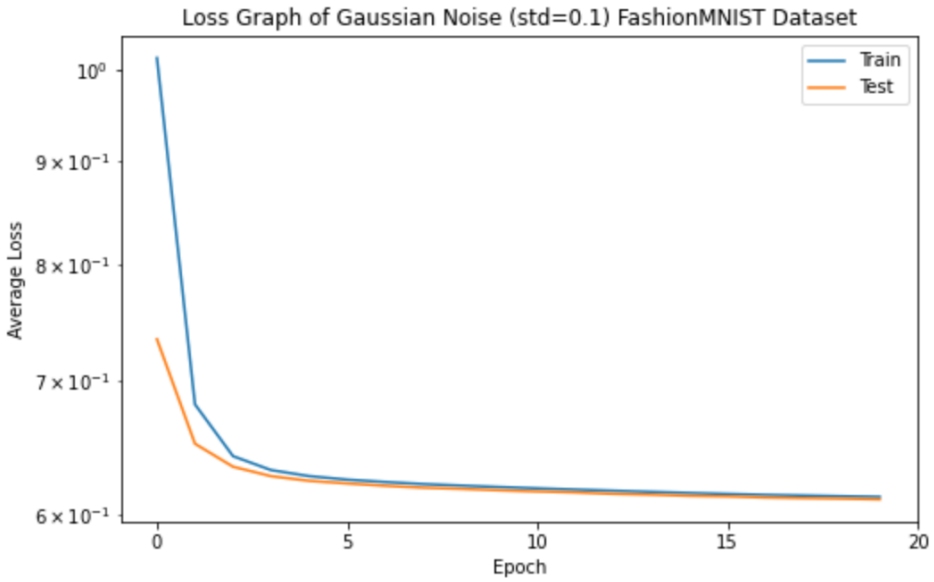

Figure 10: Loss graph of the Gaussian noise dataset (μ=0, σ=0.1)

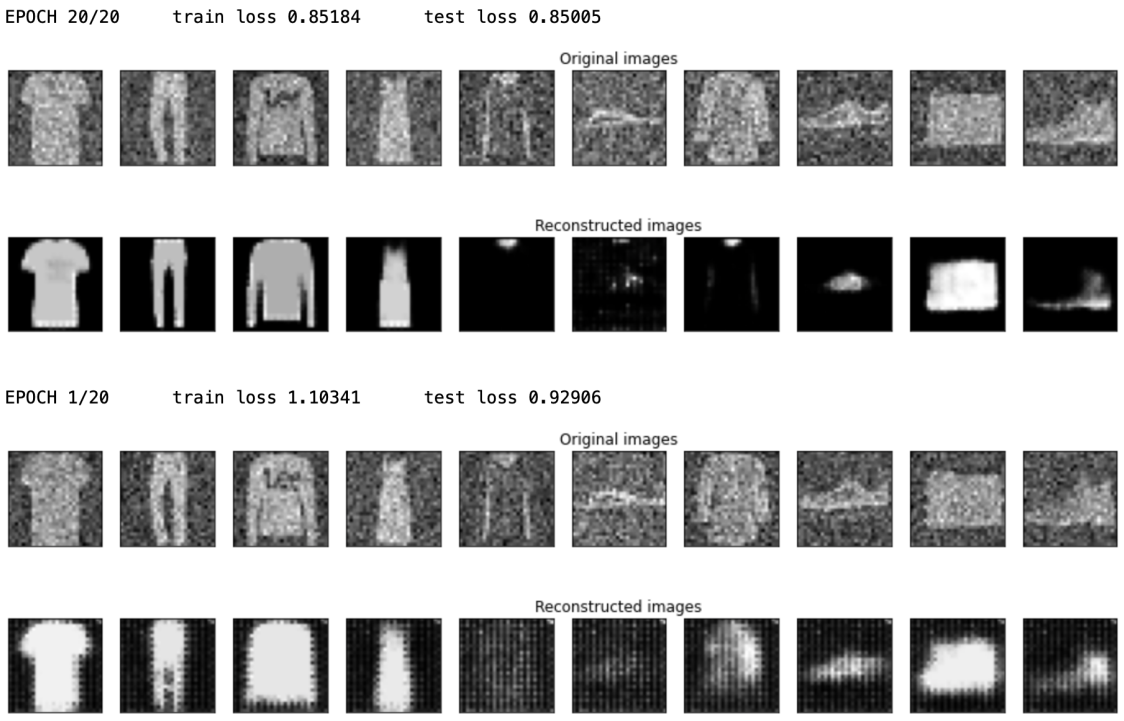

Figure 11: Train / test losses and original / reconstructed images of first and last (20) epochs of the Gaussian noise dataset (μ=0, σ=0.5)

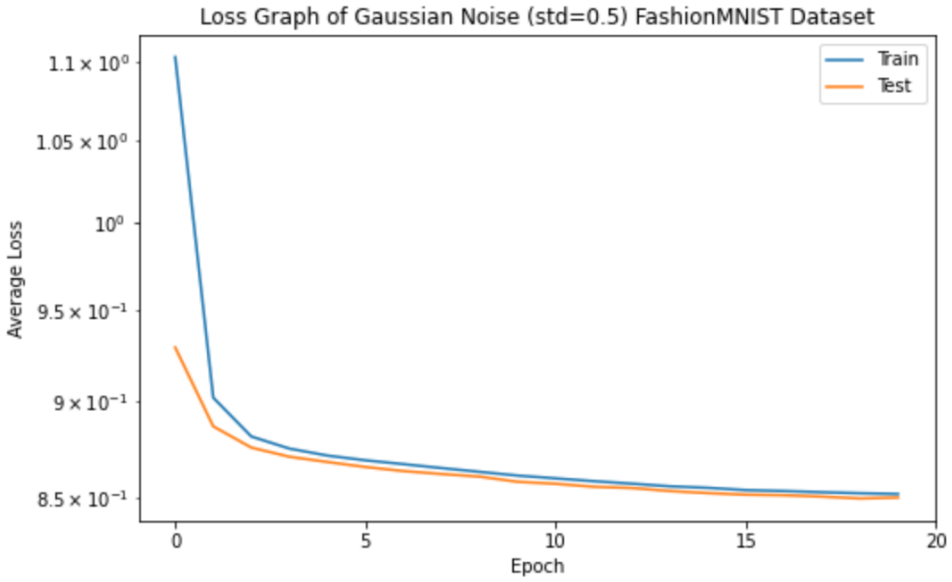

Figure 12: Loss graph of the Gaussian noise dataset (μ=0, σ=0.5)

Reference

[1] Anello, Eugenia. "Convolutional Autoencoder in Pytorch on MNIST Dataset." *DataSeries*, 28 Mar. 2022, medium.com/dataseries/convolutional-autoencoder-in-pytorch-on-mnist-dataset-d65145c132ac. Accessed 10 May 2022.

[2] Forsyth, David. *Applied Machine Learning*. Springer International Publishing, 2019, pp. 461-462, 465.

[3] Müller, Andreas Christian, and Sarah Guido. *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media, 2018, pp. 163-164