Deep Learning VAD Project Wet Part

Authors: Nativ Maor & Matan Kutz

October 27, 2024

1.3

1.3.1 Training an Auto-Decoder

1. An Overview of Auto-Decoder implementation

For the Auto-Decoder implementation we used an architecture based on the DecoderCNN of the GAN we used in HW3 part 2. The DecoderCNN was useful in generating images after it was trained as part of a GAN, so we believe the architecture has the capacity to extract useful information for classification and image generation tasks for the Fashion-MNIST dataset. The DecoderCNN architecture was based on the Decoder from the original DCGAN paper (by Alec Radford, Luke Mets, Soumith Chintala) and has proved its ability to generate realistic looking outputs from latent representations in various scenerios. It was published in a peer reviewed paper and had significant influence and reach. We took inspiration for our model parameters (like layer parameters, model depth, use of batch normalization ,activation function choices) and training parameters (optimization and tuned around them for our cases goals.

The latent space dimension we chose was 128 - not too large so it would be a trivial problem, but not too small so it would still have the capacity to hold the information for the images reconstruction.

The first layer of the network is a FC layer from the latent space to a much larger space of size 8192 (with a bias). The increase to high-dimensional latent space is typical to autodecoder as it allows for richer construction.

The second component is the DecoderCNN which is composed of 3 components of {Deconvolution layer, Batch normalization and a Relu} followed by 2 additional Deconvolution layers to fix the output size with a Sigmoid multiplied by 255 at the end of the network to make all the values of the reconstructed image to be between 0 and 255 like the expected pixel value range.

The kernel size used in the 3 convolution components is 4x4 with stride 2x2 and padding 1x1 with no bias - from 512 to 256 to 128 channels. The depth and parameter choices for the kerenel and batch normalization, and the activation function to be relu were inspired by the parameters in the DCGAN paper. The reason for batch normalization is to stabilize the training and is common practice

in training ANNs (the reasons for its effectivness are in debate but the effect of it is evident).

After these 3 Deconvolution componenets there is a deconvolution layer from 128 to 1 (1 being the channel of the generated output), followed by a 1x1 deconvolution layer with kernel size 1x1, stride 1x1 and padding 2x2 to fit the generated image size to 28x28.

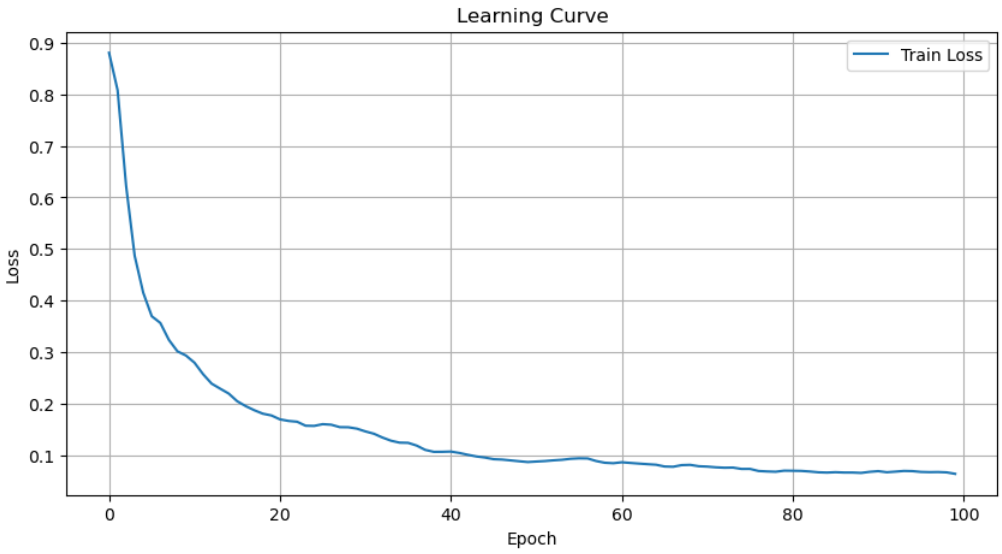
Finally, we use a sigmoid layer to fit the values of each cell of the 28x28 output to be between 0 and 1 and multiply by 255 to make the range of each cell between 0 and 255.

For the choices of the training parameters, we chose ADAM as our optimizer and used the default parameters with a learning rate of 10^{-3} (we experimented with various learning rates and found it to be the best in our search space).

To train the model we set the parameters to be optimized to be the train_latents together with the model parameters. We used the reconstruction loss provided in evaluate.py to as the loss function to be optimized.

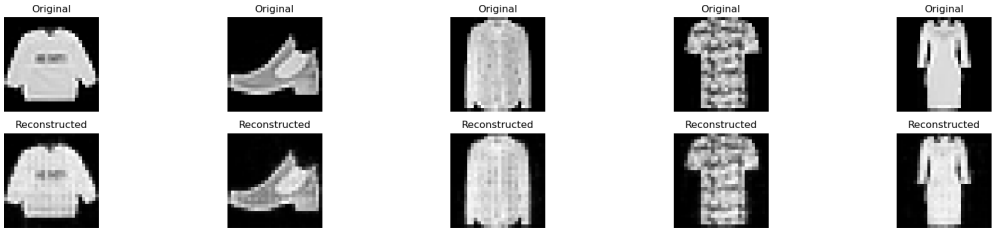
2. Evaluation of the AD model

The model training loss graph over 100 epochs:



After 100 epochs of training the reconstruction loss got down from a value around 0.9 to 0.06.

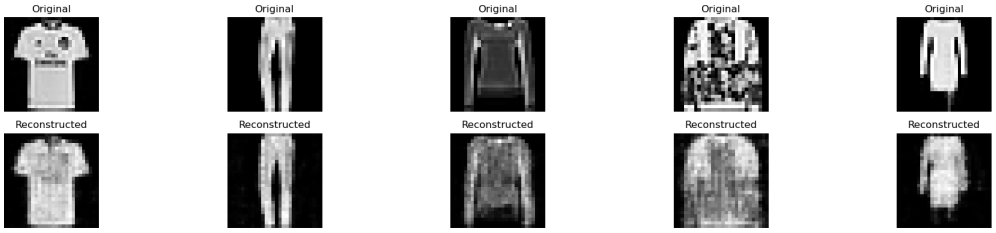
Visualization of the reconstructed images from the learned latent vectors for the train-set:



We see that the reconstruction is quite good for the training set and captures main and small features of the original images in the reconstruction.

As for the test-set, running for 100 epochs an ADAM optimizer to optimize only the test_latents with a lr of 10^-2 resulted with a reconstruction loss of 0.16.

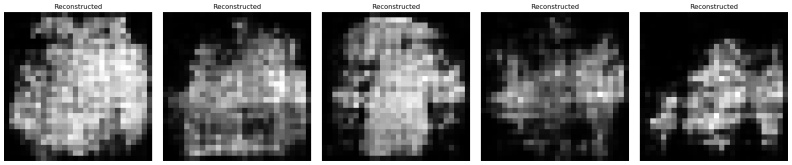
Example for reconstructed images:



We can see that the reconstruction is decent, it captures the general shape of the clothing objects. Altough it does not capture small features of the images. But it was expected for it to perform less well than the train set.

3. Sampling from $\mathcal{N}(0, I)$:

Let's observe what sampling the latent vector from a standard normal distribution will decode into,

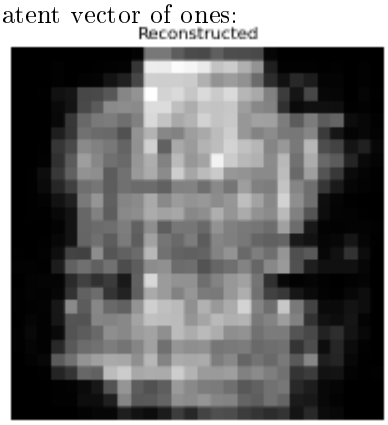


Some people might say they can recognize some similarity to clothing items in these decodings, like a heel show on the right and to its left a sneaker and further left a coat. But clearly these reconstructions are of very poor quality and are mostly wishful thinkings and our imagination knowing what to expect. It is not surprising that there will be some similarity to clothing items in the decodings, as the network was trained on clothing items, and it learned to fit vectors to it.

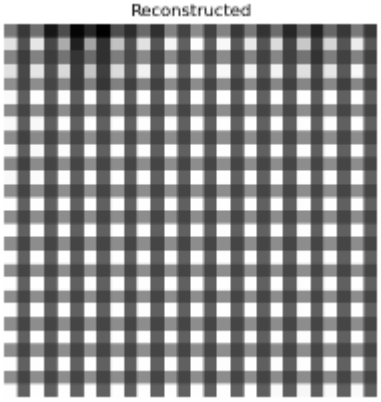
The better quality of the test cases reconstructions is resulted from the optimization on the test_latents to direct them in the direction of reconstructing something meaningful, they went through 100 epochs of training to fit as best as possible to some original image. That is while the totally random latents have nothing to train towards, there is nothing we can compare them to.

Addition, Just for fun, I tried to see what simple vectors and modification would result with:

A latent vector of ones:

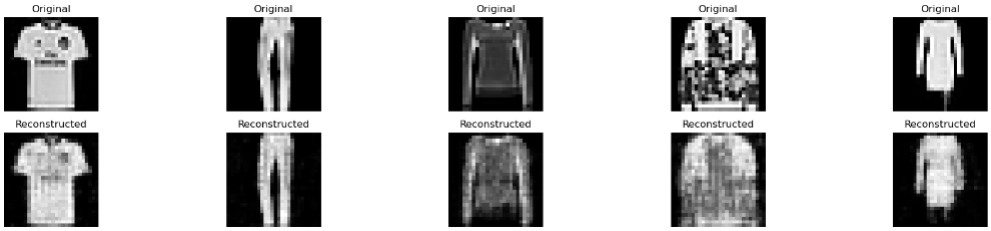


A latent vector of zeros:

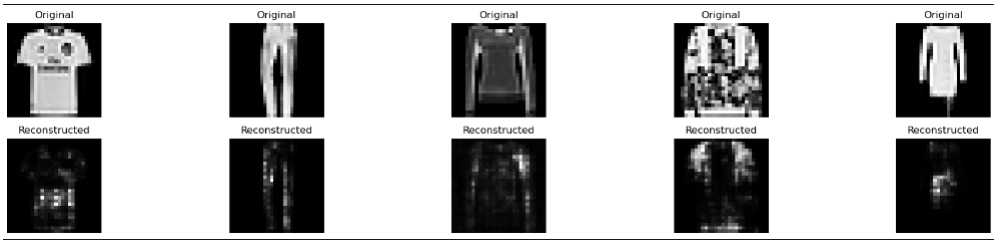


Also, while experimenting with it some more it seemed that changing only the magintude of the latent vector seemed to have an effect on the clarity of the results:

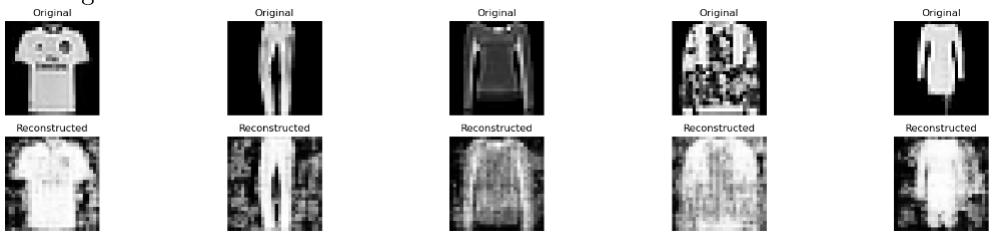
For example for the following 5 test_latents:



Half magnitude (multiplied by 0.5) results:



1.5 magnitude results:



We point this fact about the magnitude of the latents as it might be relevant in designing smarter architectures for the decoder taking the normalization of the latent vector into account.

1.3.2 Expiremienting with VAD and the latent space

1. An Overview of Variational Auto-Decoder implementation

The architecture of the VAD model is composed of the AD model, when the training and evaluation are a little different. The reason we chose to base our VAD on the AD model from the previous section is that the AD model did show some reconstruction abilities, basing the VAD on it will "isolate the variables" and allow us to learn the strength and weaknesses of the VAD approach compared to it.

To train the VAD we pass it a distribution name, and distribution parameters for each sample in the training set. We sample from a standardized distribution and compose a latent vector out of it using the distribution parameters by the reparametrization trick. We pass the latent vector through the AD model and get our output reconstruction. Upon training we optimize over the distribution parameter and the model parameters, upon evaluation we optimize only over the distribution parameters. The loss function is the ELBO loss which is composed of addition between the KL divergence term and the reconstruction loss, $loss_{total} = loss_{KL} + loss_{reconstruction}$. The model training was done over 100 epochs with ADAM optimzer with lr=10^-2 (which we found to work best after hyperparameters search) and later optimized neglecting the KL-loss for another 100 epochs as in the evaluation.

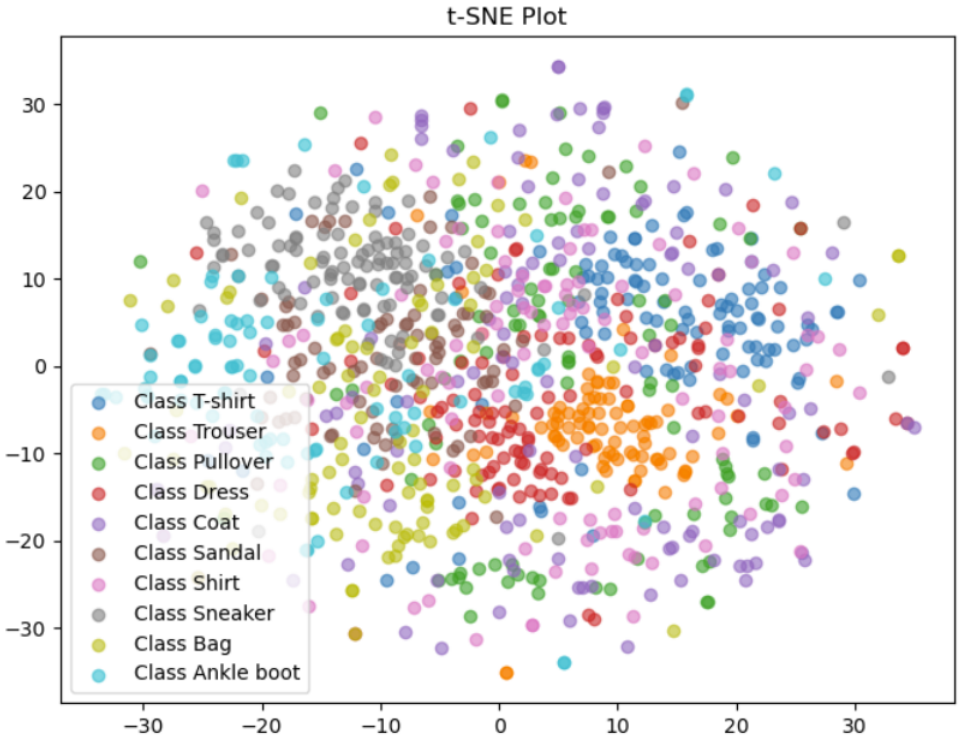
We chose 3 distributions: Guassinan, Exponential, Uniform.

For a Guassian distribution, we initialize μ and $log(\sigma^2)$ (because we want the correspondant σ^2 to be positive valued) when σ is the standard diviation. We randomize $\epsilon \sim \mathcal{N}(0, I)$ and compose a latent vector $z = \mu + \epsilon \odot \sigma$, That is

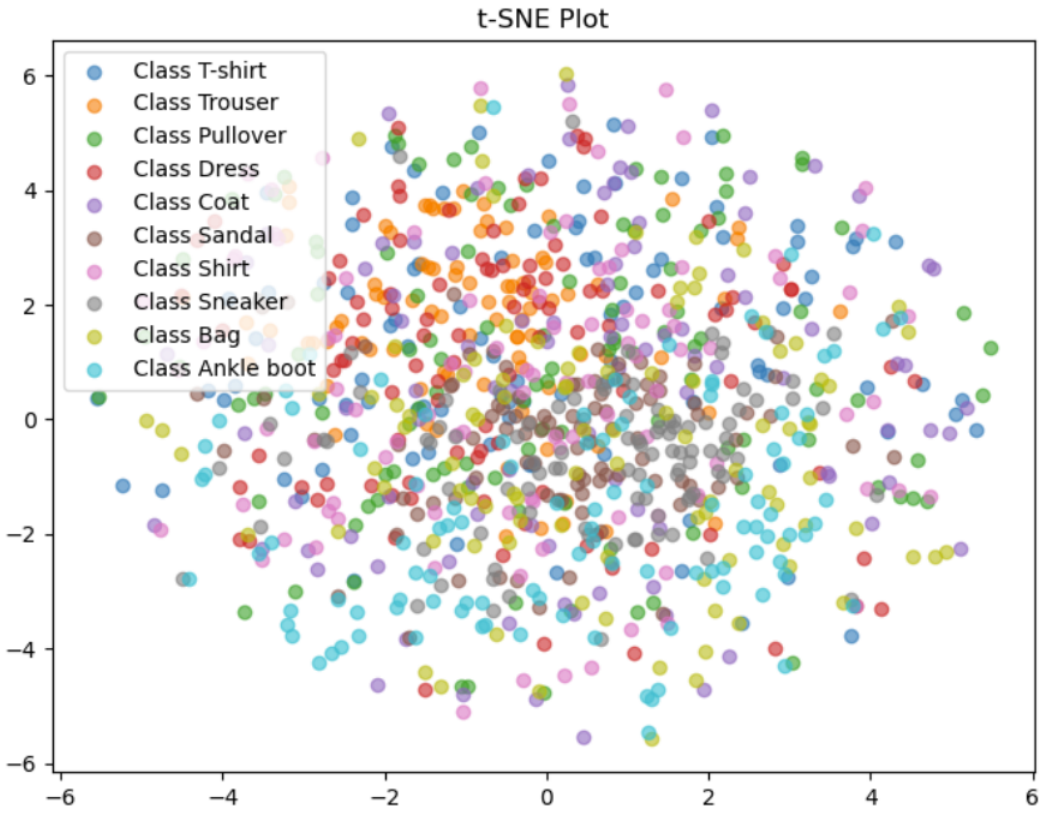
the vector that we pass through the AD model and get a reconstruction. s

2. Visualization of the latent space from the AD model using TSNE

For the train-set we get the following TSNE:



For the test-set we get the following TSNE:

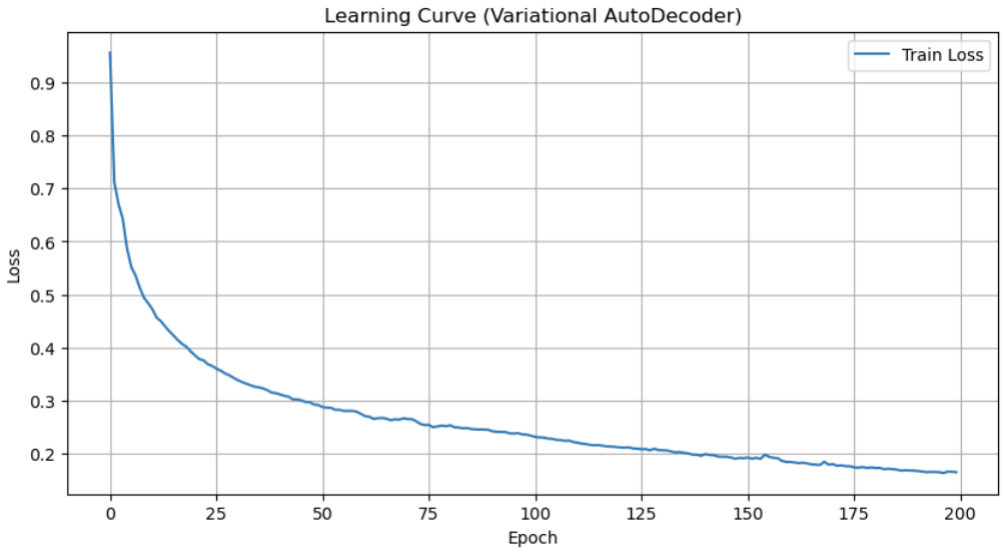


Both TSNEs are quite disorganized and no reasonable clustering of the classes occurs.

3. Evaluation of the VAD model (Gaussian)

Train

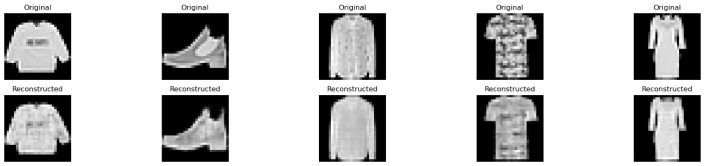
The model training loss graph over 200 epochs:



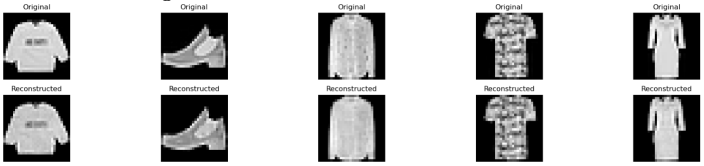
The training loss drops down to around 0.15 from an initial loss value around 1.

Right after the training, the reconstruction loss for the train-set achieved with the evaluation function is 0.116.

Sampling from the distribution learned from the learned distribution parameters for the train-set we get the following reconstructions



Evaluating 100 more epochs optimizing the distribution parameters results with the following reconstructions

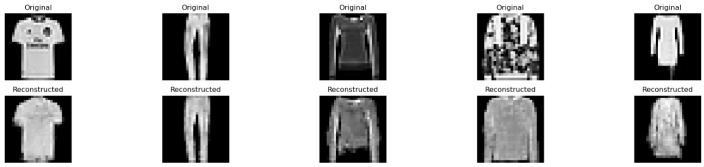


Both results seem quite good and capture the essential features, but the later results seem to capture finer details as expected.

Test

Evaluating the test-set with 100 epochs of optimization only on the distribution parameters we get a reconstruction loss of 0.22.

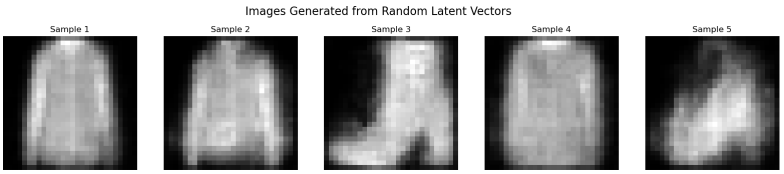
The test reconstruction results:



We can see that the overall shape is reconstructed, but fine details are lost in the reconstruction. Pretty similar to the train-set.

Randomly sampled from standard normal distribution

We sampled 5 latent vectors from $\mathcal{N}(0, I)$, decoded them and reconstructed them. These are the results obtained:

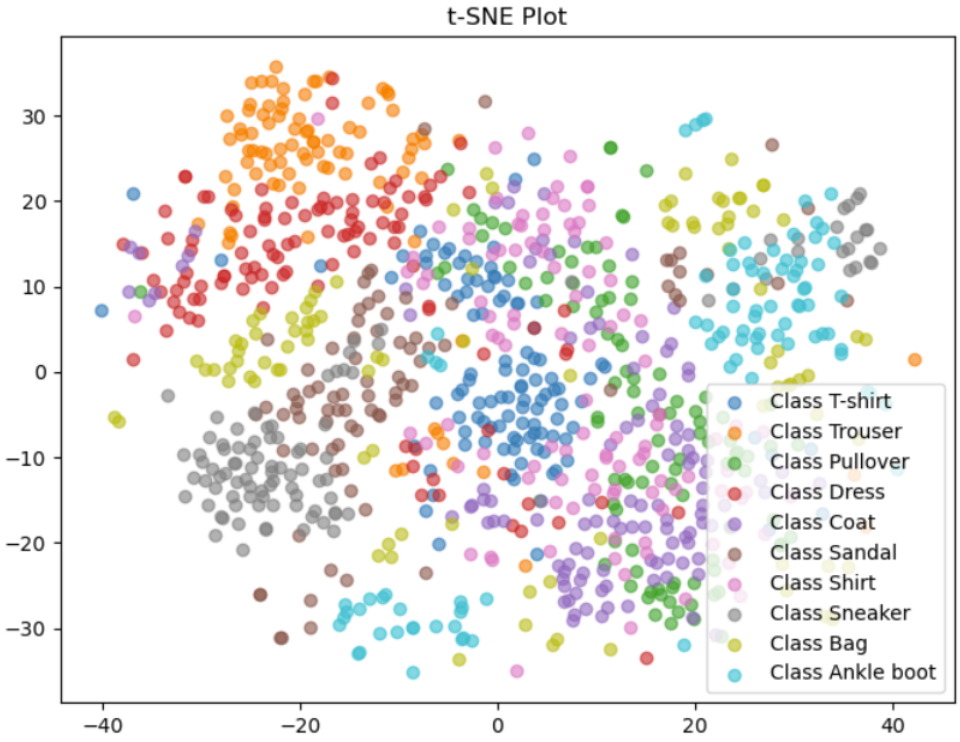


These samples look much closer to Fashion items than the reconstructions from the AD model. But the results are still poorer than the results obtained from trained test-latents, for the same reason mentioned in the previous section - they were optimized to obtain a known shape, while in the randomly sampled latents case did not. We got not too bad results, we can recognize most of the items and give a rough label to them (top clothing item or a shoe). Sample 5 is the most confusing one and is an example for the imprefection of this method to always generate something we can make sense of.

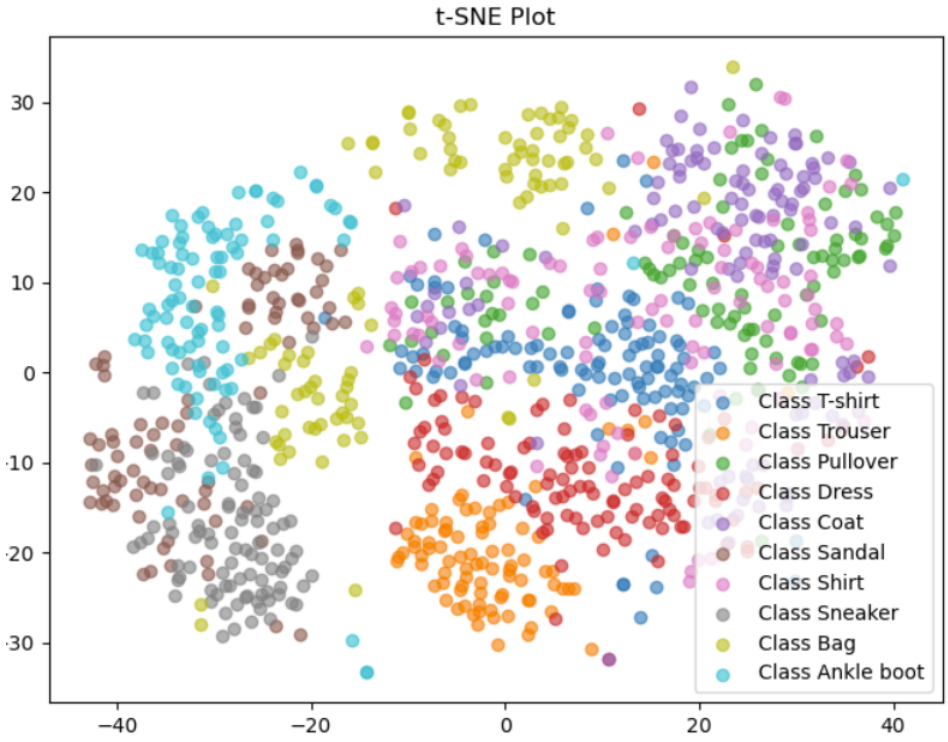
TSNE for guassian case (belongs to 1.3.2.5 but for better organization of the report i'll put it here)

TSNE plot for the train-set right after training the VAD (with the latents described by the parameters optimized):

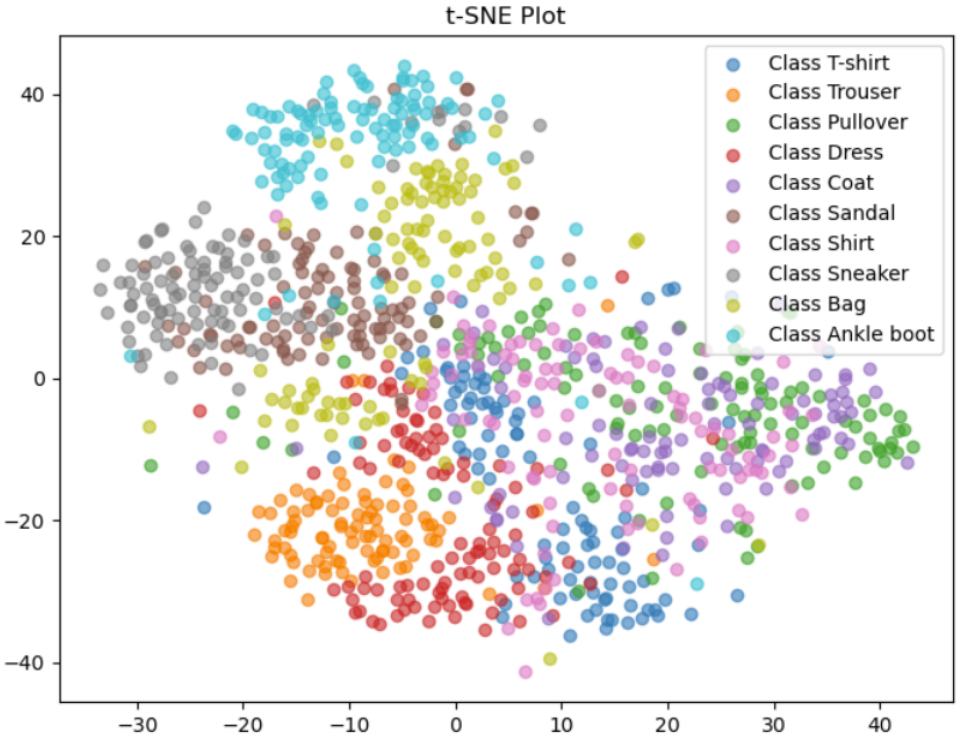
Train TSNE right after model training:



Train TSNE after 100 epochs of optimizing the distribution parameters of the train_set:



Test TSNE after 100 epochs of optimization of the distribution parameters of the test-set:



4a. Evaluation of the VAD model (Exponential)

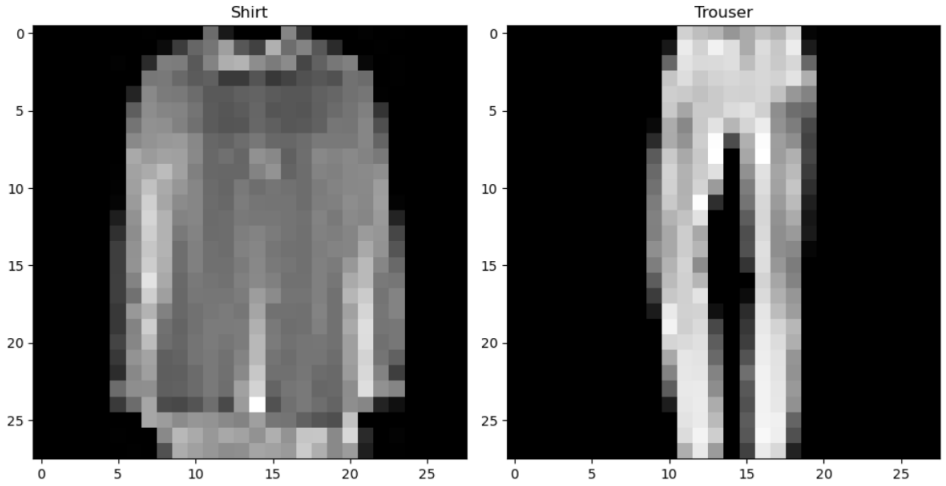
For the Exponential distribution, our parameter is $k = log(\lambda)$ and get the decay rate by $\lambda = exp(k)$ (and thus is positive for all k and will be easier to use as a parameter).

Sampling from a uniform distribution $\epsilon = U(0, I)$ the paramatrization trick for $z \sim Exp(\lambda)$ is $z = -\frac{1}{\lambda} log(\epsilon)$. this can be seen by inverting the relation of ϵ and z and get $\epsilon = e^{-\lambda z}$

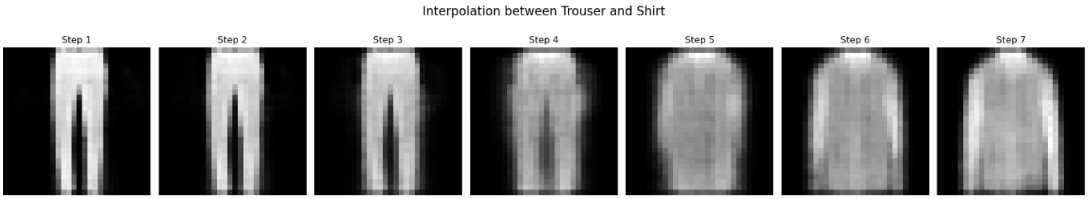
4b. Evaluation of the VAD model (Uniform)

6. Interpolations

The interpolations were done between a Trouser (1) and a Shirt (7). We expect to see something similar to a dress between them (as can be somewhat expected from the tsne view, as the bulk of the trousers are somehwere in between them).



Gaussian case -



We can "imagine" reconstructions of steps 5 and 6 somewhat resemble a dress as expected.

Exponential case -

Uniform case -

We interpolate between