Università degli Studi di Padova
NEURAL NETWORKS AND DEEP LEARNING 21/22

Homework II

Unsupervised Deep Learning

Student: *Ali Bavarchee(1219425) - Ali.Bavarchee@email.com*
Instructors: *Dr. Alberto Testolin, Dr. Umberto Michieli*

1 Introduction

The subject of the homework 2 is implementing - and in some extent applying the unsupervised deep learning.for this purpose, an autoencoder network is hired and implies over the familiar Fashion MNIST to extract its features. The input raw data is Fashion MNIST with 70k samples as 28×28 gray-scale images which are fractured to training set with 60k(split to train set with 42k and validation set with 12000 samples) and test dataset with 10k samples.

As known, the Autoencoder network is a type of ANN which is used to learning unlabelled data by encoding and compressing it (dimensionality reduction) to learn the important properties and latent attributes of input by observing it's environment and afterwards, decoding the training data to original form as the output.

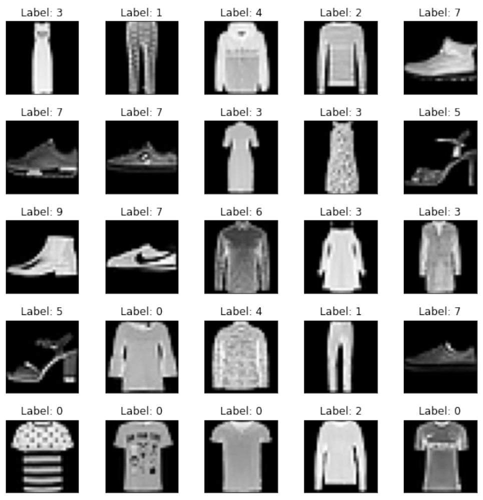

Figure 1: Visualizing some samples of input data

2 Method

2.1 First attack- Autoencoder

To extract the important features of input data - to execute the latter intents like denoising, classification, etc, one could use Autoencoder network which comprises two parts, encoding and decoding. After loading and transforming the train set, now it is the time to define two classes. Encoder class has a model function contains a fully connected convolutional network with 3 layers. The size of each image in raw data(FASIONMNIDT) is 28 x 28, so the fist layer reduces the image size to 14 x 14, next layer 8 x 8 and the last Conv layer downsizes it to 3 x 3. Then the compressed image transforms to a flatten form, the mini- batch with 256 converts to a vector with 73728 components, and then with 2 linears layers, the output is compressed to 3. This means that 256 x 1 x 28 x 28 sample data is compressed to a vector with 3 components which with other sample data forms a whole mapped data with compressed form so called latent space. In latent space, the samples melt together, so the ”distance” is reduced and it is much more easier for machine to distinguish each important features of data, as the hidden similarities and differences reveal in latent space. But it is not possible to use these important features, untill executing the next part, decoding. The decoding part is the inverse function of encoding, mapped the compressed data from latent space to form(and size) of orginal data and the acme of this part is extracting the original images. Like encoding part, class ”decode” do this job. First by two linear layers, each point

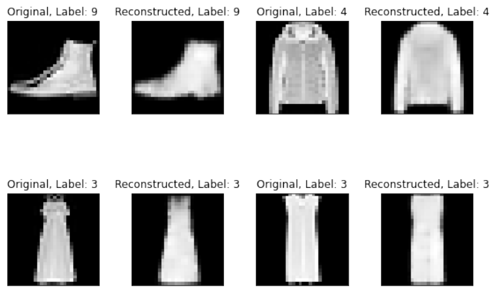

Figure 2: Original vs. Reco samples

in latent space resize to a vector with 73738 elements, and the 3 convolutional layers, deconvolve the vector-form samples to original image size. The next job is to initialize the network, which is already done. The loss function is selected meas squared error function, based on the input data(matrix form which each element represents the lightness or darkness of a pixel in the image) and the model(undesupervised). Adam optimization algorithm (as the single optimizer) and learning rate is 0.0005. Then the training function (train_epoch) and testing function (test_epoch) as pre-train and pre-test are defined to train and test a single epoch. Then we can do iteration by using for loop to reach the optimal results. It is obvious that the more times(epoches) of training makes the better results, but it also consumes more time. The number of epoches here is 10 same as lab 05. Some samples of the original image and regeneration image are shown in figures.

2.2 Dimensionality Reduction by PCA T-SNE

As mentioned before, Autoencoder sends the data to hidden space by reducing the size of data and recovers them to find out the important hidden features. There are another methods to compress data like PCA and T-SNE as the famous ones. To show and investigate the ”distance” of each encoded sample in latent space, it is used PCA and TSNE precooked functions (from sklearn) which are demonstrated in figs.

2.3 Transfer learning

To improve and upgrade the model, the results of the previous homework (classification task of FASIONMNIDT) is used. First, start by defining models(a convolutional neural network with 3 hidden layers and then reload the network state. After that, data is sent to training loop to train again by fine-tuning. The confusion

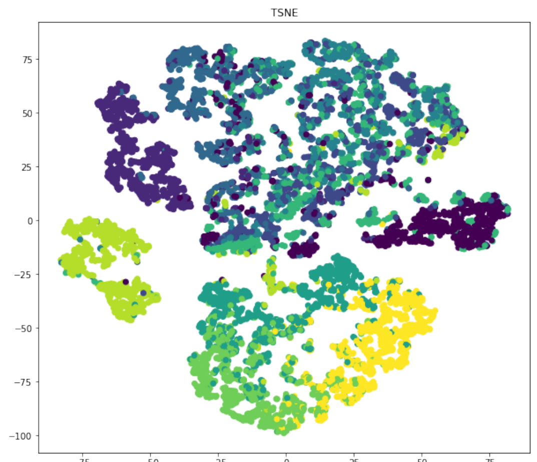

Figure 3: TSNE of latent feature space(10 classes)

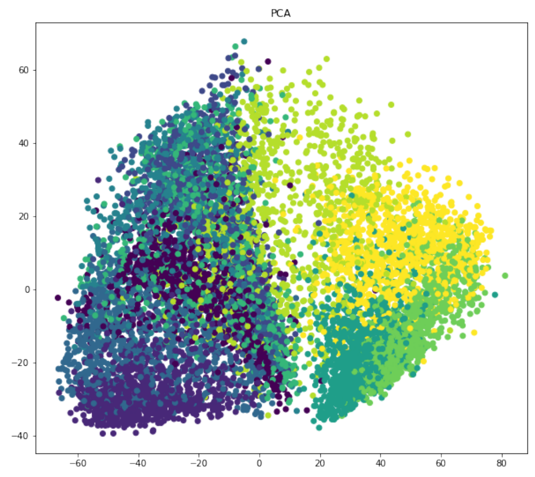

Figure 4: PCA of latent feature space(10 classes)

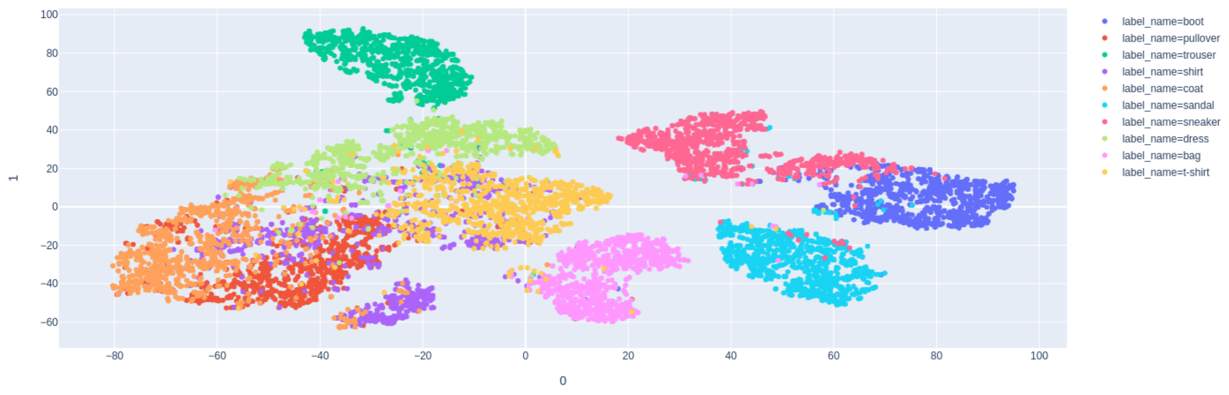

Figure 5: Plotting latent feature space by TSNE

matrices of before and after fine-tuning are reported in fig. Also TSNE and isomap embedding of the feature space of the dataset are shown at fig.

2.4 VAE

Variational Autoencoder is another method in unsupervised class to dig out the important features of rae data. In this method, the original images are smeared with some noises (Gaussian noise) and the sent them to encoder-decoder function, but results should be the noise free original samples while the inputs are noisy. It is a good practice for machine/learner to master the data!

2.5 Latent space exploration

Surveying of latent features space and visualization of its vector (surface) assists to reach the better model which could be more sensitive to changes in an arbitrary factors. It is possible by changing the hyperparameter , and then plotting the encoded data by PCA and T-SNE.

2.6 Hyperparameter tuner assistant - Optuna

In fact the input data that is used here is FASIONMNIDT which is known(it has 10 classes and the it is somehow familiar to writer!), but in general, the data is not docile as FASIONMNIDT and selecting hyperparameters could be a troublemaker. To tune a hyperparameters one can use some tuners. Optuna

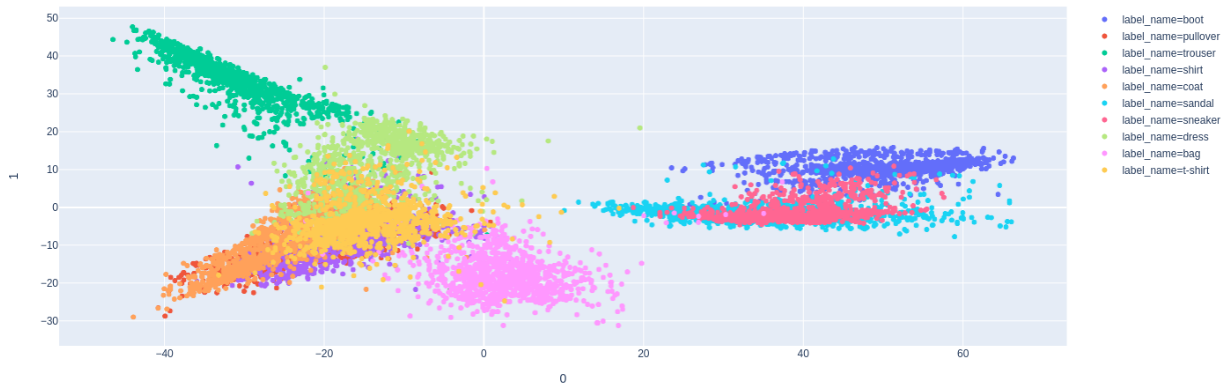

Figure 6: Plotting latent feature space by isomap embedding

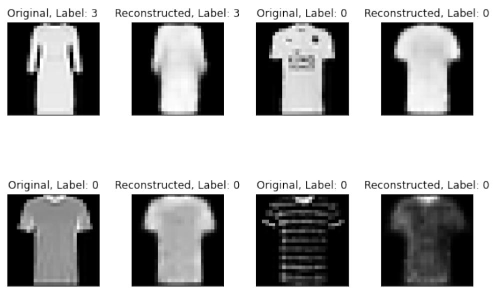

Figure 7: Some examples of VAE performance

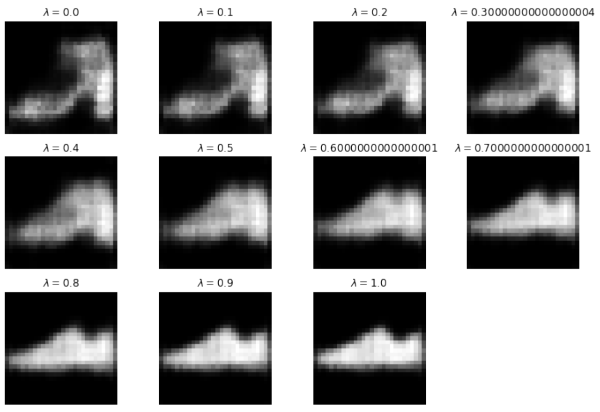

Figure 8: Reco of a random sample with different

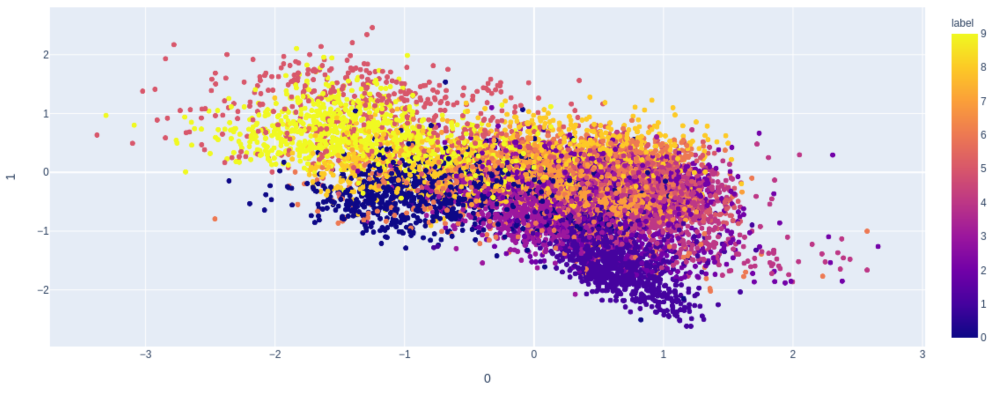

Figure 9: Scatter plot of latent space

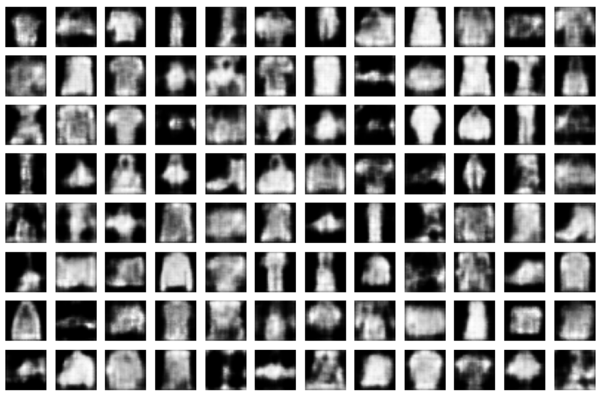

Figure 10: An instance of enceded sample data by VAE

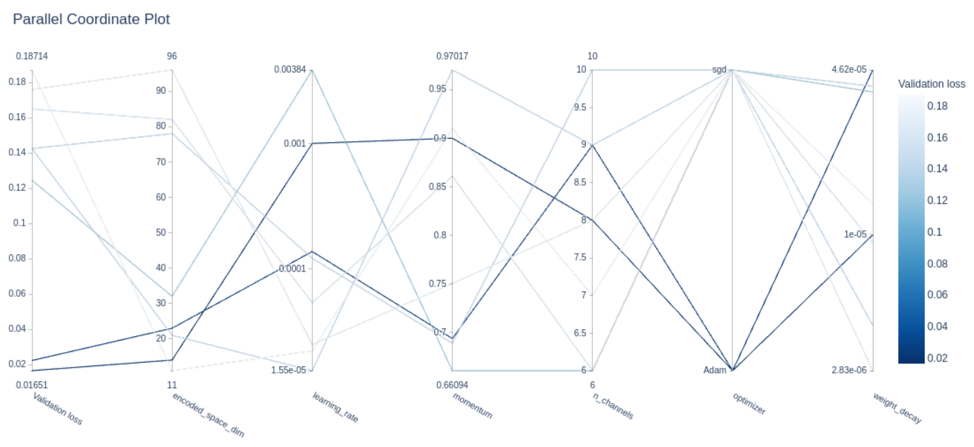

Figure 11: Weights which are studied by call back function under Optuna

actually is an automatic hyperparams optimization software framework which is compatible with Pytorch. Also it works with Pytorch Lightning library which provides a high level interface and powerful framework for Pytorch.

3 Results

The Autoencoder results is pretty acceptable. It is improved by using previous homework results and fine tuning. The VAE also is comparable with Autoencoder. Finally data is train by Autoencoder, but in this time using tuner to tune the model by the best hyperparameters and as it is observable in the figure of some random original and reco samples, the performance of the model is satisfying. Although the train loss of Autoencoder reached to around 0.015, and it is the least train function without further processes, it has a difference with test loss and the later processes such as VAE and transfer learning are ment to extenuate this difference and suppress the overfitting.

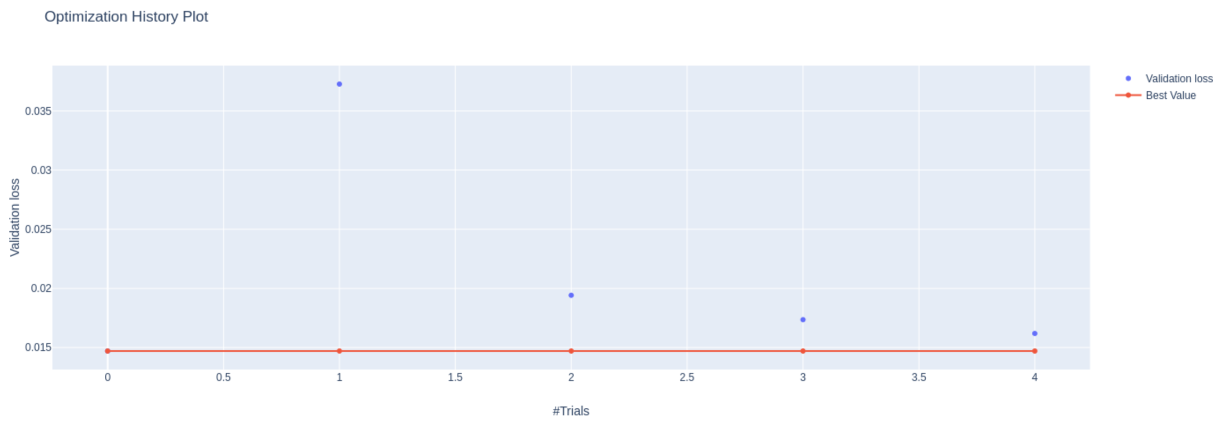

Figure 12: History of optimizarions conduced by Optuna

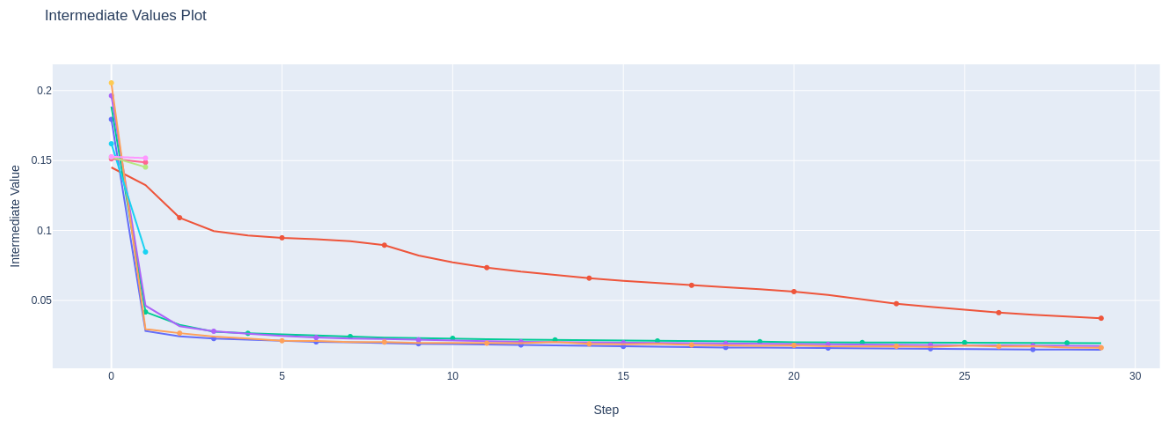

Figure 13: 30 trails to find optimize hyperparams

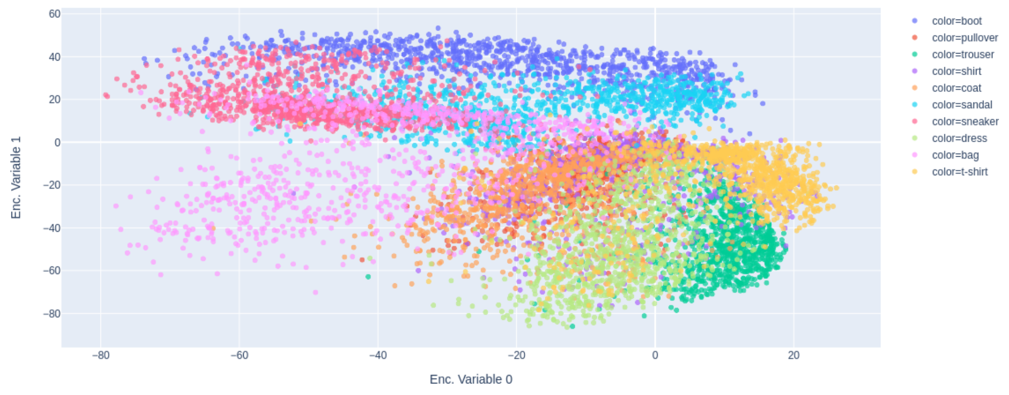

Figure 14: Visualization of latent space of Aoutoencoder

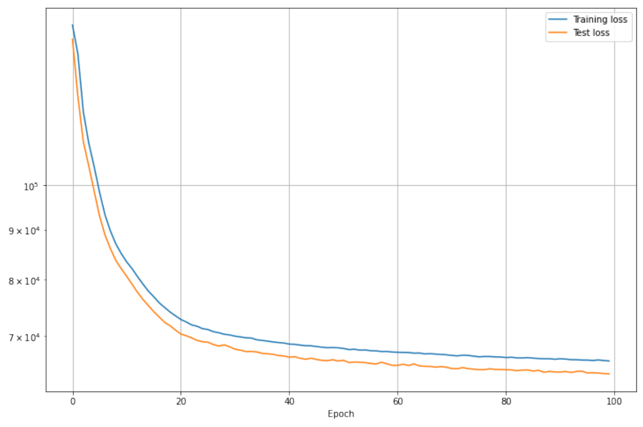

Figure 15: Train/Test loss function - VAE model

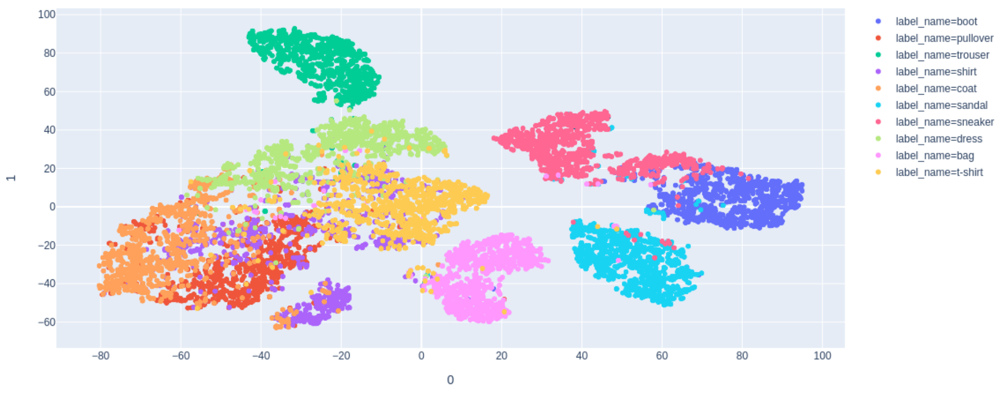

Figure 16: TSNE of enceded samples after Fine-Tuning

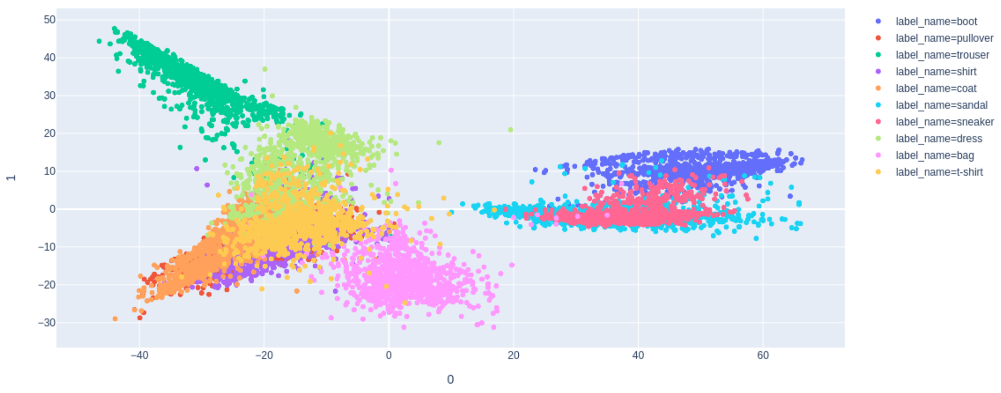

Figure 17: PCA of enceded samples after Fine-Tuning

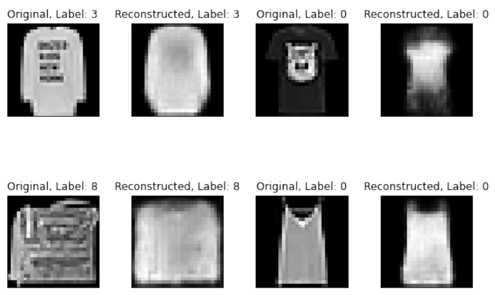

Figure 18: Performance of the best model tuned by Optuna

,,