# Anomaly Detection on FashionMNIST

Alessia Pontiggia, 1892079

08 Sep 2023

## 1 Introduction

Anomaly Detection (AD) consists in identifying observations in a dataset that significantly deviate from the remaining observation.
Most of the times, datasets are not labeled since anomalies are rare and can differ in shape and type. This leads to unsupervised learning techniques.
Once the model has been built, AD is about classifying data as Normal (i.e. withous anomalies) or Anomal so it is a binary classification task.

## 2 Repo Github and Technology

The full project is available at Github Repo containing the dataset and code in a Google Colab Notebook. The technology used to build and train the models is **Pytorch Lightning**. The code is written to run on the Google Colab (free version) provided **GPU**.

## 3 FashionMNIST Dataset

The set of data used is the **FashionMNIST** dataset made up of 60.000 training samples (50.000 for training set, 10.000 for validation set) and 10.000 test samples. Each sample is a grayscale image of 28x28x1 shape with an associate label from 0 to 9 representing the category of clothes it belongs to.

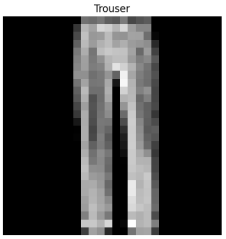


Figure 1: Trouser

# 4 Solutions

In the following sections possible solutions are described and developed.
The first approach gets inspired by the baseline "*Robust Anomaly Detection in Images using Adversarial Autoencoders*" [1].
The second approach follows the "*Auto-Encoding Variational Bayes*" pipeline [2].
The last approach is implemented following the current State-Of-The-Art algorithm *"GAN-based Anomaly Detection in Imbalance Problems"* [3].

# 5 Baseline1: Convolutional AutoEncoder, Loss and KDE

The first baseline uses the convolutional autoencoder (AE) to detet anomalies on the fashionMNIST dataset. The intuition behind this approach is using two parameters as anomaly scores: the **reconstruction loss** (MSE) and the **kernel density estimation** (KDE) in an Autoencoder model.
The autoencoder model uses unsupervised learning and neural networks to find the deepest relations within data, represented into the lower dimensional vector z of the input x in order to reconstruct the input $\hat{x_i}$ as close as possible to x.
The reconstruction loss for a sample i is the squared difference between the reconstructed image $\hat{x_i}$ and the original image $x_i$.

$L = (\hat{x_i} - x_i)^2$

The autoencoder tries to minimize this loss L.

The kernel density estimation gives an estimate of the Gaussian kernel distribution over the space for each embedding $z$ produced by the encoder with a fixed bandwith $h$.

$K(z; h) = e^{-}(\frac{z^2}{2h^2})$

The key idea is to exploit two important facts given by the AE structure:

1. The loss of anomal (normal) samples should be bigger (smaller) than the mean of normal (anomal) samples loss.
2. The KDE of normal (anomal) samples should be very close within them, instead anomal (normal) samples should have a KDE far from that "normal (anomal) space distribution".

At this point, determining if a sample is anomal or normal means thresholding its loss and its kde.

## 5.1 Convolutional AE: architecture

The fundamental architecture is a Convolutional Autoencoder which uses in the Encoder the convolution operation to downsample the input into the latent representation and in the Decoder the transposed convolution operation to upsample the embedding to the original dimensions.

Convolutions are used because images have lots of parameters and with just few layers a reduction of nearby 13 times the initial number of parameters is obtained (each starting image has 784 parameters while the embedding contains only 64 elements). Also, convolutions preserve spatial properties of the images and extract important features automatically.

### 5.1.1 Encoder

The encoder maps the input into single values, which will compose the embedding vector z, by learning the relations within data. We can refer to this process as a learnable function $f_\phi$. The encoder has three convolutional layers each followed by the relu activation function. The final embedding is a 1x1x64 vector.

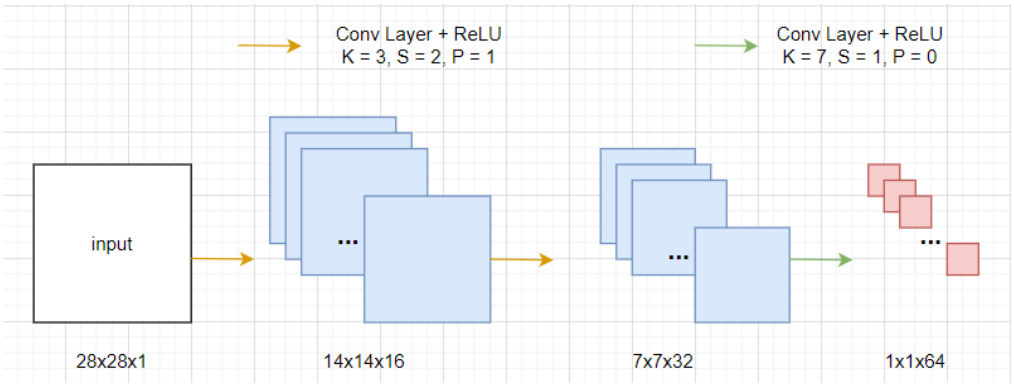


Figure 2: Encoder structure

### 5.1.2 Decoder

The decoder tries to reconstruct the input using its embedded representation z. We can refer to this process as a learnable function $g_\theta$. The decoder follows the inverse procedure of the encoder by applying two transposed convolutional layers each followed by the relu function and a third transpose convolutional layer followed by the sigmoid function.

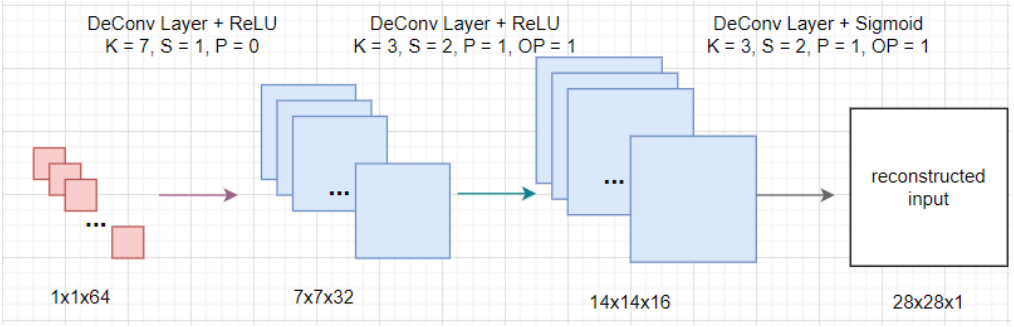


Figure 3: Decoder structure

### 5.1.3 AutoEncoder

The resulting autoencoder combines the downsampling process of the Encoder and the upsampling process of the Decoder together.

## 5.2 Training, Validating and Testing

The learning process wants to minimize the reconstruction loss (MSE) between ouputs and inputs:
$minL = min(\hat{x_i} - x_i)^2 = min(f_\theta(g_\phi(x)) - x)^2$
This means finding the best $\phi$ and $\theta$ parameters that define $f_\phi$ and $g_\theta$.
During the training, we assume that each sample is normal. The training process is done over the training set through a maximum number of **100 epochs**. The **Early Stopping** regularization technique monitors the validation loss and stops the learning process when this metric doesn't decrease its value for at most 3 times. The goal of this step is minimizing the MSE Loss (i.e. the reconstruction error between the original image and the one reconstructed by the decoder).

| Training Set | Validation Set | Test Set |
|---|---|---|
| 50000 (512) | 10000 (256) | 10000 (256) |

| Learning Rate | Weight Decay | Max Epochs | Regularization |
|---|---|---|---|
| 0.0001 | 0.00001 | 100 | Early Stopping |

## 5.3 Tensorboard outputs

### 5.3.1 Loss Plots

Loss plots are taken from TensorBoard Logger that are stored during the training and validation processes.

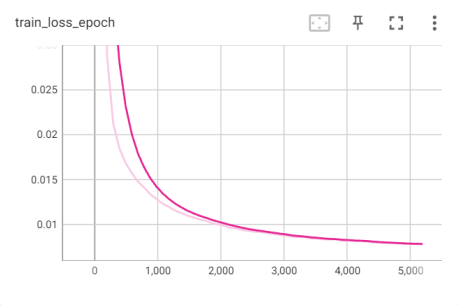


Figure 4: Training Loss

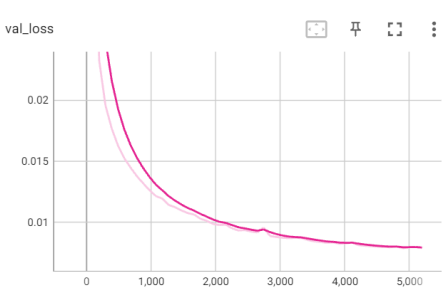


Figure 5: Validation Loss

As the graphics suggest the process converges to a very small loss value within 52 epochs:

- **train_loss** = 0.00782
- **validation_loss** = 0.00787
- **test_loss** = 0.0079

#### 5.3.2 Reconstructed Images

Left image reports the input image while right image reports the reconstruction of each input. They belong to the test set.

Figure 6: Input images

Figure 7: Reconstructed images

### 5.4 Detecting Anomalies: MSE and KDE

In this part, an anomal class has to be choosen. The label "5" (sandals) will be the category of clothes defined as anomal. The other part of the dataset is normal. In this case the anomalies occur 6.000 times over 60.000 i.e. 10% of the starting dataset contains anomalies.

#### 5.4.1 LOSS and KDE over normal and anomal samples

The MSE loss is calculated for each normal sample (and then anomal sample) of the training set and stored. Then, the means are taken.

KDEs have to be calculated over only normal samples embeddings of the training set first (and then only anomal ones). This allows to understand the area where normal samples (and anomal samples) are distributed according to their kernel density in order to define thresholds. Once KDE is calculated and stored for each (normal and anomal) sample, the means are taken separatedly.

### 5.4.2 KDE and Loss Distributions

In order to understand how the means of normal and anomal loss and kde are distributed, a plot of them is reported in the images below (since they are scalars the y axis is fixed as 0).

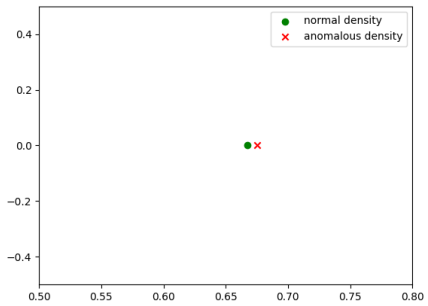


Figure 8: Normal and Anomal average densities

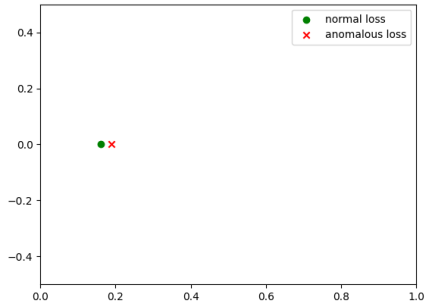


Figure 9: Normal and Anomal average losses

### 5.4.3 Thresholds

In order to exploit properties of loss and KDE explained at *Section 8*, a threshold value needs to be found. The validation set (containing both normal and anomal samples) is used for this and the threshold value which produces the biggest number of correct classifications is kept.
The value of the threshold is choosen by verifying the following condition:
If the density of the sample i is "far away" *enough* from the average of the "normal" ("anomal") density AND if the loss of the sample i is far away *enough* from the average of the "normal" ("anomal") loss, then it will be classified as an anomal (normal) sample. Enough is what the threshold value defines.
A single threshold value is used both for KDE and Loss because, as the *Figure 8* and *Figure 9* show, the distance among the two is almost the same.

## 5.5 Results and Metrics

Once all discussed before is done, a test function for a binary classification task is implemented. The results are reported with some metrics, including the ROC curve:

- Accuracy: 99%
- Precision: 99%
- Recall: 98%
- F1-Score: 49%

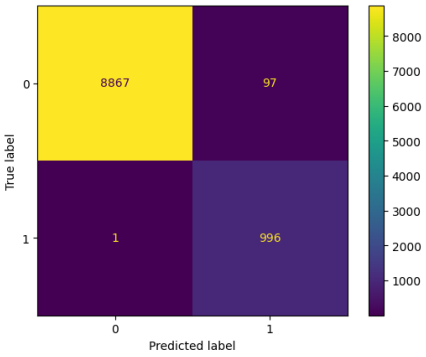


Figure 10: Confusion Matrix

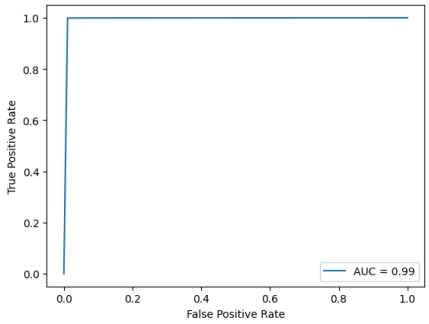


Figure 11: AUROC

# 6 Baseline2: VAE

This second baseline uses the Variational Autoencoder (VAE) to detect anomalies on the fashionMNIST dataset.
The anomaly score is given by the **reconstruction loss** (BCE): anomal samples should have a bigger loss w.r.t normal samples. Again, a threshold value has to be found.

## 6.1 VAE: architecture

Such as the AE architecture does, VAE consists in encoder and decoder branches.

VAE provides a method for jointly learning deep latent-variable models and corresponding inference models using stochastic gradient descent.
In this process, x is our data and we assume it is generated from a hidden variable z which follows a Gaussian distribution.

### 6.1.1 Probabilistic Encoder

The Probabilistic Encoder maps the input x into an approximated (Gaussian) distribution q(z|x) which we want to be as close as possible to the real posterior distribution p(z|x) that generated data. We can not directly get P since its computation is intractable.
Rather than outputting values (points) for the latent state as we would in a standard autoencoder, the encoder model of a VAE will output two vectors describing the mean $\mu$ and variance $\sigma$ of the latent state distributions q.

The encoder structure has two convolutional layers followed by the ReLU function and two Linear layers that will output $\mu$ and $\sigma$. In details:

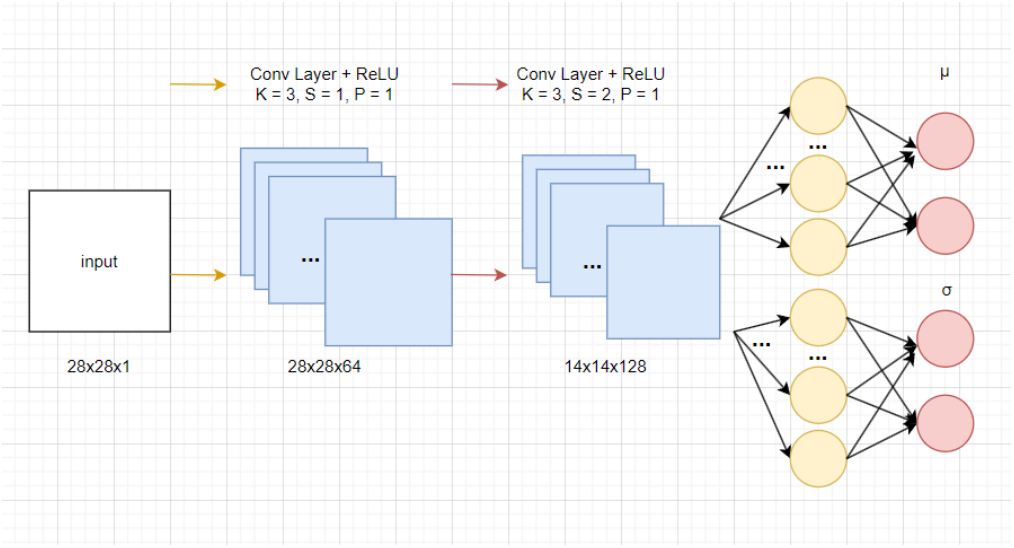


Figure 12: Probabilistic Encoder structure

From this distribution q(z|x) described by $\mu$ and $\sigma$ we should sample z, which will feed the decoder. Since the sampling process is stochastic and backpropagation can't be applied over it, we use a reparametrization trick which suggests that we randomly sample $\epsilon$ from a unit Gaussian, and then shift the randomly sampled $\epsilon$ by the latent distribution's mean $\mu$ and scale it by the latent distribution's variance $\sigma$. In formulas: $z = \mu + \sigma * \epsilon$, where $\epsilon \approx (0,I)$.
Since $\sigma$ can assume negative values, in the formula above $\sigma^2$ is used.

### 6.1.2 Probabilistic Decoder

The Probabilistic Decoder model will proceed to develop a reconstruction of the original input by finding the distribution p(x|z).

Its structure is made up of a linear layer followed by the ReLU function and two transposed convolutional layers, the first one followed by the ReLU and the second one followed by the Sigmoid function. In details:

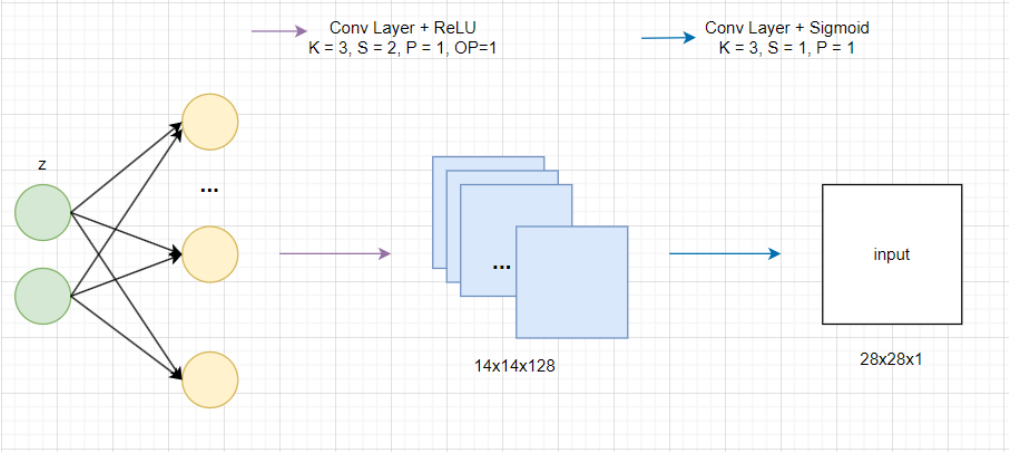


Figure 13: Probabilistic Decoder structure

### 6.1.3 VAE

The variational autoencoder combines the probabilistic encoder and decoder output.

## 6.2 Training, Validating, Testing

The learning process is about:

- finding the best Q (the one closer to P) to generate samples similar to the real ones i.e. minimizing the KL Divergence between Q and P: **KL(q(z|x) || p(z))**
- minimizing the reconstruction loss between generated samples (outputs) and the real samples (inputs): $E_z$ **log p(x|z)**

These two steps are summed up in a unique loss, called ELBO, defined as:
L = $E_z$ log p(x|z) - KL(q(z|x) || p(z)).
The **Early Stopping** technique is used and monitors the validation loss with patience set to 3. The maximum number of epochs is set to **100**. The training is done through only the normal training set, so that the reconstruction loss of unseen sample will be higher. Validation and test sets contain both normal and anomal samples.

| Training set | Validation Set | Test Set |
|---|---|---|
| 50000 (512) | 4000 (256) | 10000 (256) |

| Learning Rate | Max Epochs | Regularization |
|---|---|---|
| 0.0001 | 100 | Early Stopping |

## 6.3 Tensorboard outputs

### 6.3.1 Loss Plots

Loss Plots are taken from Tensorboard logger during the training process.

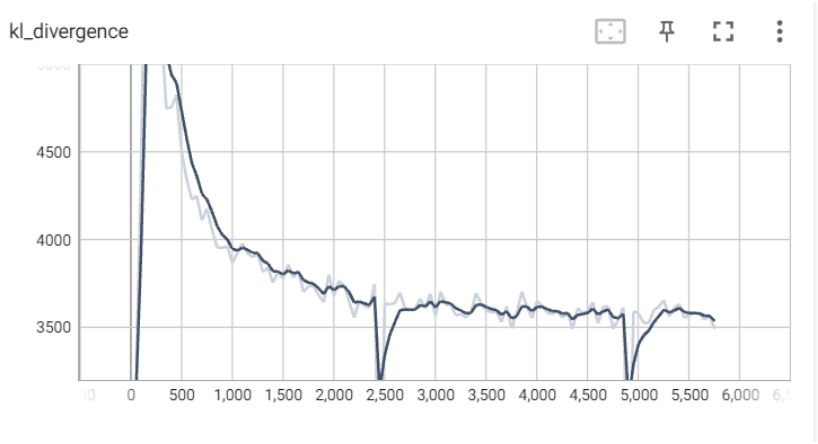


Figure 14: KL Divergence

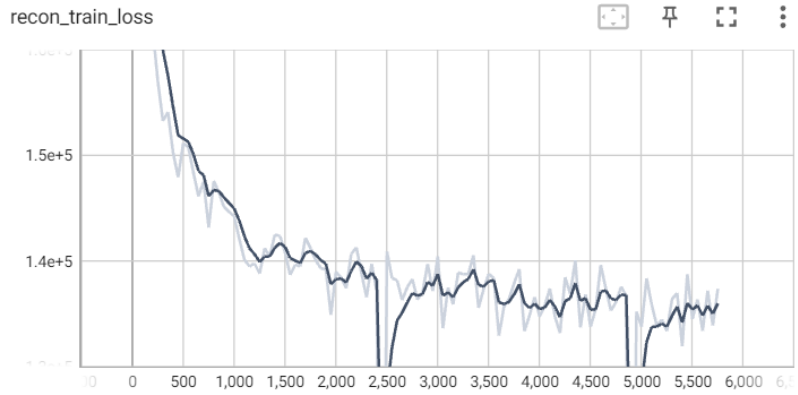


Figure 15: Reconstruction Loss

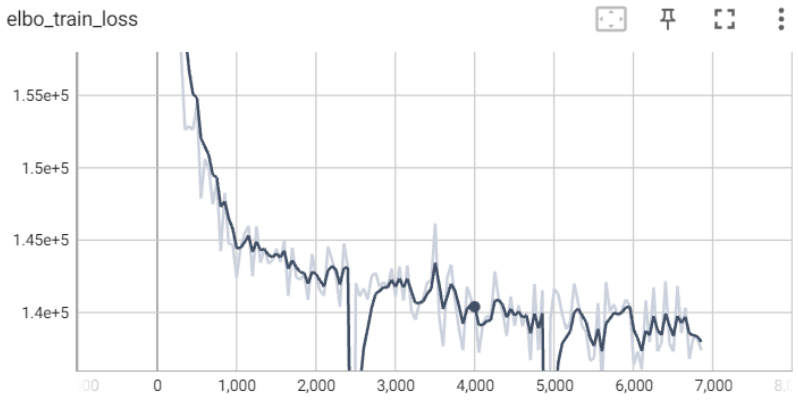


Figure 16: Elbo

### 6.3.2 Input Images vs Reconstructed Images

Left image reports the input image while right image reports the reconstruction of each input. They belong to the test set.

Figure 17: Input images

Figure 18: Reconstructed images

## 6.4 Detecting Anomalies

In this approach we distinguish the normal dataset and the anomal dataset. As before, the class choosen to be anomal is the "sandals" class (label n.5). Validation and test sets contain both anomal and normal samples. In order to detect anomalies, the reconstruction loss (BCE) is used.

### 6.4.1 Loss over normal and anomal samples

To define a threshold value, the loss is computed for each sample of the normal validation set and anomal set (remaining from the dataset without normal samples). The means are taken as threshold values.

### 6.4.2 Thresholds

Defining the anomal average loss as AL and the normal average loss as NL, the condition to classify a sample x as

- **normal** is *loss(x_hat, x)* less than AL
- **anomal** is *loss(x_hat, x)* greater than NL

## 6.5 Results and Metrics

At this point a test function is built to sum up the results for a binary classification task.

Here the metrics:

- Accuracy: 92.95%
- Precision: 96%
- Recall: 90%
- F1-Score: 46.5%

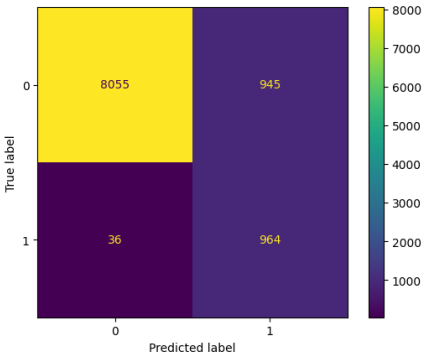


Figure 19: Confusion Matrix

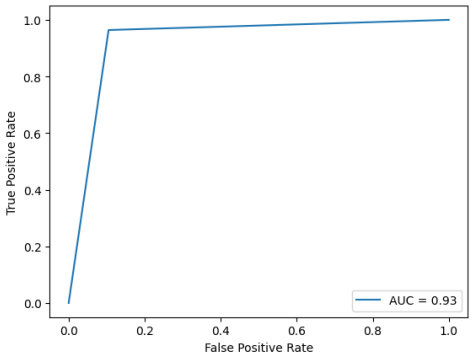


Figure 20: AUROC

# 7 SOTA Algorithm

In this case, the implementation and decisions taken are related to the paper where the current State-Of-The-Art for Anomaly Detection on FashionMNIST dataset is discussed.
Some modifications are introduced due to technology limitations and missing information. They are all specified (with ***bold and italic formats***) in the next sections.
The intuition behind this final approach is to maximize the reconstruction error when the anomal data is entered to the generator and to minimize the reconstruction error when the normal data is entered to the generator.

## 7.1 Class Imbalance: Remodule FashionMNIST Dataset

The first problem presented in the paper is related to the class imbalance of anomalies with respect to "normal" images. For this reason, there is one normal class (at each experiment) and the other nine classes are treated as anomal. In this implementation just ***one experiment*** [1] has been done choosing the label 0 to be normal and the others (from 1 to 9) to be anomal. Since the normal class contains 6000 samples, from the other 9 anomal classes are picked some ***randomical*** samples [2] so that the total number of normal and anomal samples is quite the same.

- Training Set: 9600 samples
- Validation Set: 2400 samples
- Test Set: 2000 Samples

Differently to the paper implementation, here ***it's not used data augmentation and images are not resized.***

[1] In the paper 10 experiments are conducted so that each class is normal once
[2] In the paper the anomal samples are taken using K-Means Clustering technique

## 7.2 GAN: architecture

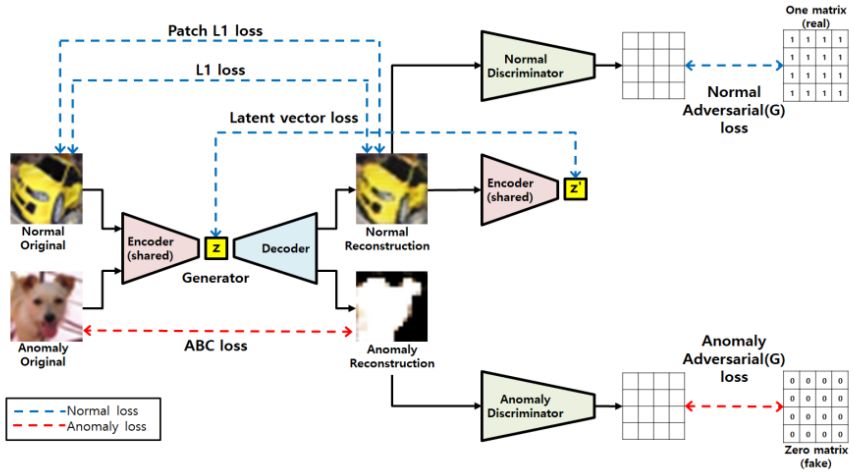


Figure 21: GAN architecture

### 7.2.1 Generator

The Generator is a convolutional autoencoder ***which structure is slightly different*** to the one proposed in the paper, in terms of kernel dimensionality and number of convolutional (and transposed convolutional) layers. [3]
It is made up of:

**Encoder**:

1. **Convolutional Layer** Output Dimensionality=32, K=3, S=2, P=1, followed by ReLU
2. **Convolutional Layer** Output Dimensionality=64, K=3, S=2, P=1, followed by ReLU
3. **Convolutional Layer** Output Dimensionality=128, K=7, followed by ReLU

**Decoder**:

1. **Transposed Convolutional Layer** Output Dimensionality=64, K=7 followed by ReLU
2. **Transposed Convolutional Layer** Output Dimensionality=32, K=3, S=2, P=1, OP=1, followed by ReLU

[3] In the paper the generator is a symmetric network that consists of four 4 x 4 convolutions with stride 2 followed by four transposed convolutions.

3. **Transposed Convolutional Layer** Output Dimensionality=1, K=3, S=2, P=1, OP=1, followed by Sigmoid

#### 7.2.2 Discriminator

As the *Figure 21* shows, the architecture involves two discriminators, one for the anomal images and the other for normal images. This wants to solve the discriminator distributional Bias problem explained in the paper as the situation that occurs when *"the generator is trained to output 1 from normal data and 0 from anomaly data (...). Thus when training from both normal and anomaly data, using a single discriminator results in training the model to classify only normal images well"*. The structure is the same for both discriminators and it is slightly different to the one used in the paper [4]:

1. **Convolutional Layer** Output Dimensionality=32, K=3, S=2, P=1, followed by ReLU
2. **Convolutional Layer** Output Dimensionality=64, K=3, S=2, P=1, followed by ReLU
3. **Convolutional Layer** Output Dimensionality=128, K=3, S=2, P=1, followed by ReLU
4. **Convolutional Layer** Output Dimensionality=1, K=3, S=2, P=1, followed by Sigmoid

### 7.3 Losses

To achieve the purpose explained at *Section 7*, the proposed method uses the weighted sum of 6 types of loss functions to train the generator, one for normal discriminator and another for anomaly discriminator.
To solve the *"loss function scale imbalance"* problem, each loss has its own weight so that the effect on the learning is different.

#### 7.3.1 Discriminator Loss

Both discriminators use a-b coding scheme where a and b are the labels for fake data and real data, respectively:

$$min_D VLSGAN(D) = [(D(x) - b)^2] + [(D(G(x)) - a)^2]$$

[4] In the paper the discriminator is a general network that consists of three 4 x 4 convolutions with stride 2 followed by two 4 x 4 convolutions with stride 1

### 7.3.2 Generator Losses

All six losses used to train the generator are here reported. The first four are employed for normal images, the last two are employed for anomal images.

1. L1 reconstruction error of generator:
   $L = ||x - G(x)||_1$
2. Patch loss [5]
   $L = f_a(n) = (||x_1 - G(x_1)||_1), i = 1, ..., M$
3. Latent vector loss
   $L_e = ||E(x) - E(G(x))||$
4. Adversarial Loss
   $min_G VLSGAN(G) = [D(G(x) - 1)^2]$
5. Adversarial Loss
   $min_G VLSGAN(G) = [D(G(x) - 0)^2]$
6. ABC loss used to maximize L1 reconstruction error for anomaly data
   $LABC = -log(1 - e^{-L_\theta(xi)})$

## 7.4 Training, Validating and Testing

The training is done using [6], [7], [8]:

| Training Set | Validation Set | Test Set |
|---|---|---|
| 9600 (1) | 2400 (1) | 2000 (1) |

| Learning Rate | Weight Decay | Max Epochs |
|---|---|---|
| 0.0001 | 0.00001 | 10 |

[5] In the paper the Patch Loss uses an average over n=3 patches based on the loss scores, in this implementation all the M=16 patches are used

[6] In the paper 300 epochs are used

[7] In the paper lr = 0.0001

[8] In the paper lr decay factor = 0.5

### 7.4.1 Input Image vs Reconstructed Image

After training, the results (i.e. the reconstructed images) are taken from the test set. Here some of them are reported:

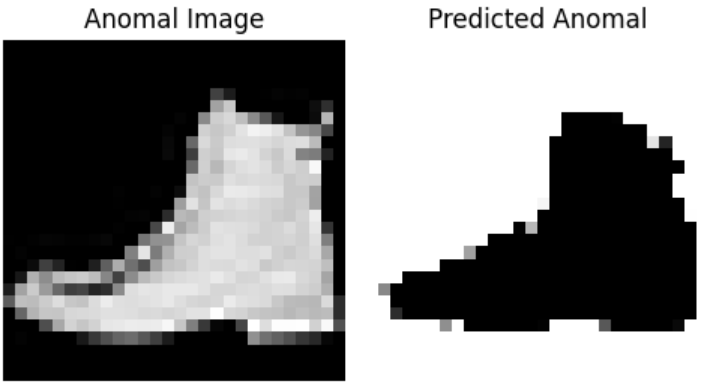


Figure 22: Anomal input reconstruction

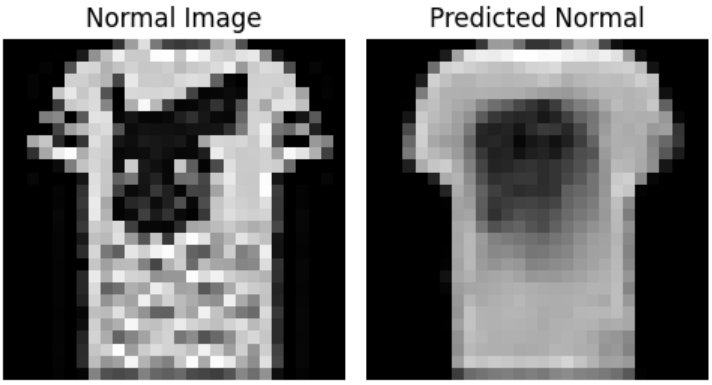


Figure 23: Normal input reconstruction

## 7.5 Detecting Anomalies

As explained before, the Gan has been trained to minimize the reconstruction error when normal data is entered to the generator and to maximize the reconstruction error when anomal data is entered to the generator.
Thus, the anomaly score here is the reconstruction loss.

### 7.5.1 Threshold

In the validation set a threshold is computed as the difference between the mean of reconstruction loss of normal samples and the mean of reconstruction loss of anomal samples. After several experiments, the best threshold results to be 0.5.

## 7.6 Results and Metrics

- Accuracy: 94.85%
- Precision: 98%
- Recall: 91.7%
- F1-Score: 94,74%

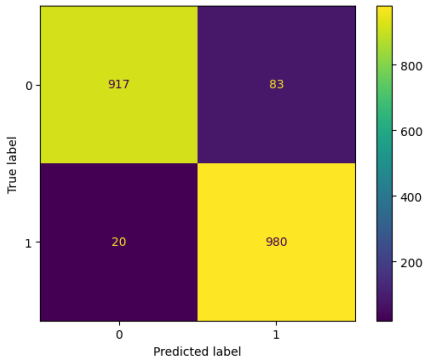


Figure 24: Confusion Matrix

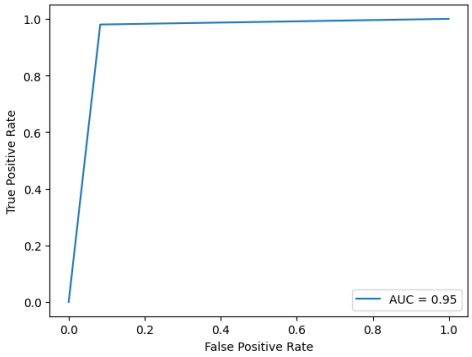


Figure 25: AUROC

# 8 Conclusions

In this project the problem of anomaly detection on FashionMNIST Dataset has been analized.
Three different implementations have been proposed: AutoEncoder (baseline 1), Variational AutoEncoder (baseline 2) and GAN (SOTA model).
The baselines show good results: they are almost infallible in classifying normal images as normal but fail more in classifying anomal images as anomal. This may be caused by the threshold value and also by the imbalance between normal and anomal samples.
The implementation of SOTA model tries to solve this last issue by remoduling the dataset. In fact, the F1-Score is way bigger than the ones produced by the baselines.

The table below reports the requested metric (AUROC) for each of the proposed solution:

| AE | VAE | GAN |
|---|---|---|
| 0.99 | 0.93 | 0.95 |

# References

[1] Robust Anomaly Detection in Images using Adversarial Autoencoders,
*Department of Statistics, Ludwig-Maximilians-University Munich, Munich, Germany.*

[2] Auto-Encoding Variational Bayes

[3] GAN-based Anomaly Detection in Imbalance Problems,
*Department of Electronic Engineering, Seokyeong University, Seoul, Korea*