UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Digital Forensics A.Y. 2021-2022

Final Project: Anomaly Detection and Classification

Nicoletti Gianpietro 2053042
University of Padua:
ICT for Internet and Multimedia
(Cybersystems)

Abstract

This project involves the implementation of different types of Anomaly detection strategies (Isolation Forest, One-class SVM and autoencoder) and the preprocessing performed to improve the performances of the models. After the anomaly detection a classification of the anomalies was performed.
For all the models the accuracy was taken as evaluation of the performances.
This report contain only the results obtained and some important observations without explanation of the theory below the strategies used.

---

1 Introduction

Given a dataset of grayscale images of 256x256 pixels describing the status of a network[1], the objectives of this project are two:

- Distinguish the normal images from the abnormal images;
- Classify the type of anomaly.

The implementations of the models were done in Python using the *Scikit-Learn* library for the machine-learning strategies and *Keras* library for the neural-network strategies.

2 Anomaly Detection

In order to perform an anomaly detection, various methodologies were implemented on subsets of the train set and test set. This was done to reduce the computational time and to avoid the saturation of the main memory. In particular 7000 and 3000 samples were extracted randomly respectively for the train and the test set. Notice that, in the code attached to this report, there is the possibility to use a different number of samples and also to decide whether to use a pre-computed results or to create from scratch the models.
The trainings of the models were made with the assumption of clean data. Instead the test sets contain samples from different classes: *clean* (normal data) and *dos11, dos53, scan11, scan44* (abnormal data).
In Figure 1 the distribution of the samples is shown.

[1] https://www.frontiersin.org/articles/10.3389/frsip.2021.814129/full

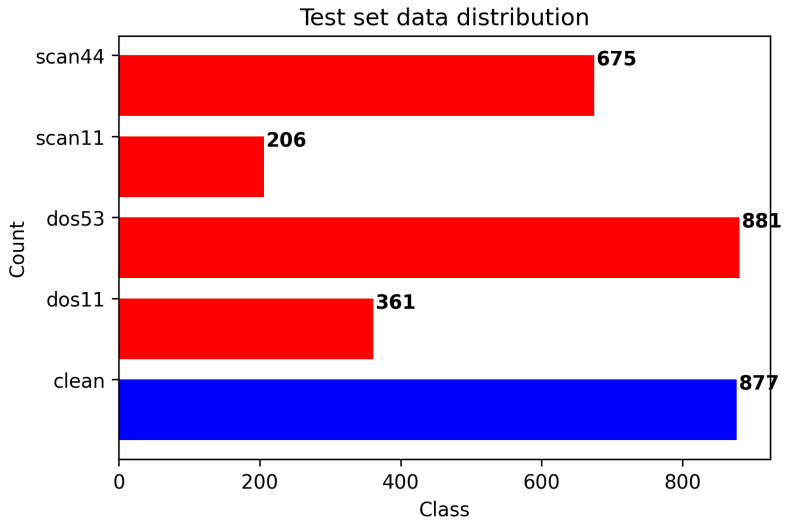

Figure 1: Distribution of the data in the test set (anomaly detection)

2.1 Isolation Forest and One Class SVM

For both models the implementations in *Scikit-Learn* library were used and preproccessing of the images was not performed. As expected the combination of the few samples and the huge amounts of features in the images (256x256 = 65536) led to bad performances during the test.
In particular for the Isolation Forest, using 50 trees the accuracy is about 100% and 29.23% in the train and test set respectively.
For the One Class SVM, trained with SGD, the performances were slightly better: 99.5% (train set) and 42.03% (test set).
Various tests were performed with different values of the parameters but nothing brought particular differences to the results obtained. For this reason in this section the tests are not reported.

2.2 Improvement of Performance

Notice that, at this point, only two things can be done:

- Increasing the number of samples in order to try to avoid the over-fitting of the models or try to increase the number of trees for the Isolation Forest;
- Find a way to reduce the number of features.

In the project only the second option was investigated for two main reason: it keeps a small computational time and it permits to create a predictor that can be trained with few data. In particular the last point is very important since, in a realistic scenario, it is not always possible to use a huge amount of data or, better yet, there is not always a big number of samples.
To reach the second point nine features were extracted from the images:

- Sum of the values inside the pixels
- Count of the number of black/not black pixels
- Count of the rows/columns with only black pixels
- Min/Max index of a not "only black" row/column

The idea was to obtain geometric characteristics of the images since, after a quick look at the pictures, the only difference between normal and abnormal data, is the number of "not black" pixel and their distribution

in the image.
With this features space the performances grown very much, reaching, for the Isolation Forest (with the same number of trees) an accuracy of 96.11% during the train phase and 94.47% in the test phase. Instead for the SVM there was not performance increase and, again, no results are reported.

2.3 Auto-encoder

Without going deep in the theory of the neural networks and the auto-encoders, an important observation must be done: since the images are mostly made of black pixels (as can be seen in Figure 5 in the classification part) during the reconstruction phase, made by the decoder, the most convenient way to reconstruct the images, in order to reduce the loss, is to provide in output a series of black images. For this reason the first step to do is to modify the MSE loss function to reduce the impact of the black pixels in the final result: we can define the square difference for each pixel (p) and its reconstruction($\hat{p}$) in this way:

$$(\Delta p)^2 = \begin{cases} (p-\hat{p})^2 \ , & if \ \ p \neq 0 \\ w(p-\hat{p})^2 \ , & if \ \ p = 0 \end{cases}$$

Where w is the weight of the black pixel.

With this definition of square distance and setted $w = 0.0001$, the mean was computed as definition of MSE (dividing the sum of the square differences by the total number of differences, in this case 256x256). Moreover the normalization was not performed in order to underline better the difference between "black" and "not black" pixels. After a lot of different design, the best design found is reported in Figure 2.

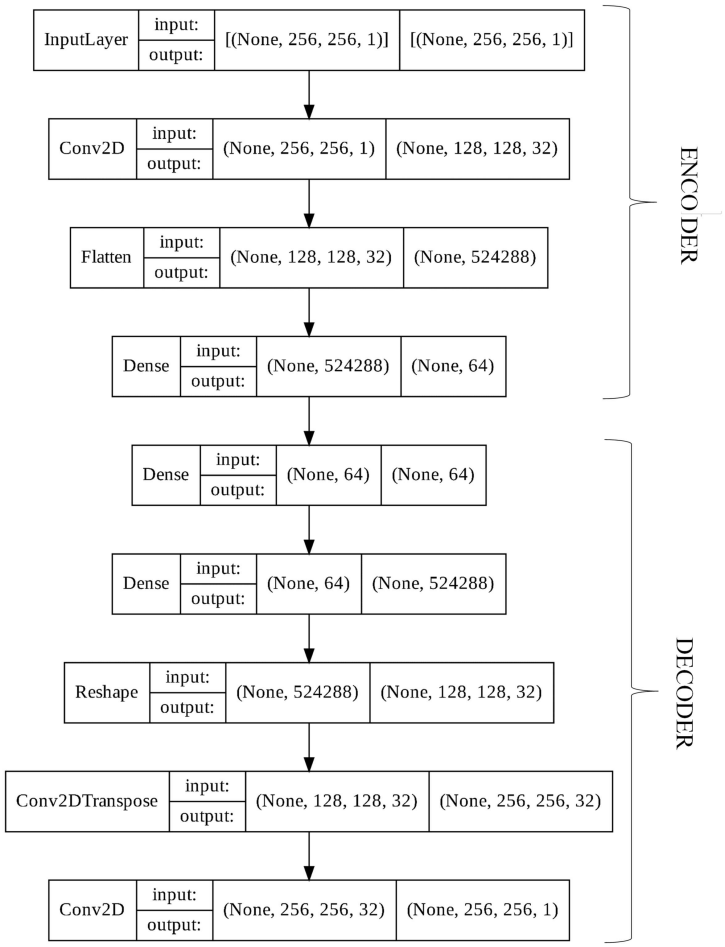

Figure 2: Design of the autoencoder

The training phase was set to stop if the value of the loss function doesn't decrease for at most 4 epochs.
With this design and setting the threshold for the reconstruction error to $\overline{CUSTOM\ MSE}+0.8\sigma(CUSTOM\ MSE)$ the accuracy is 93.88% during the train and 87.23% during the test.

3 Anomaly Classification

For the classification a new train set was generated in order to include also the abnormal data. The distribution of the data is reported in Figure 3.

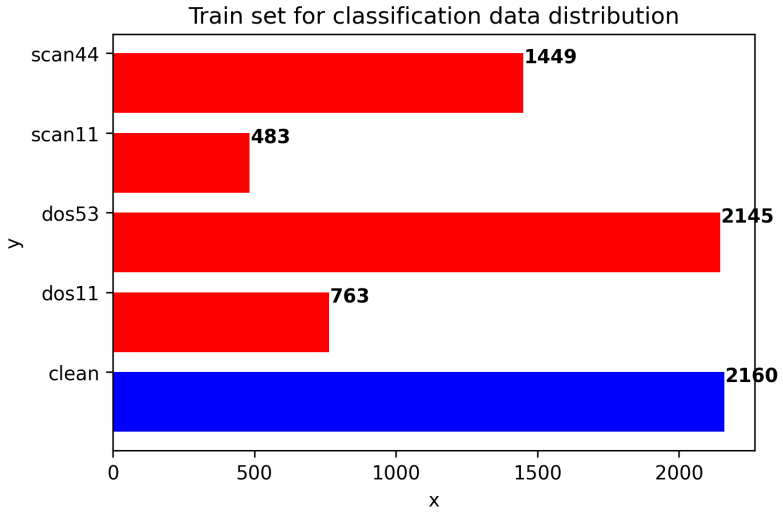

Figure 3: Distribution of the data in the train set (classification)

The assignment of this project asked to perform this part using an unsupervised technique, but after a lot of testes where clustering was implemented using the raw data, the encoded data (with the encoder implemented in the section 2.3) and the features space extracted in the section 2.2 with poor performances, the code from the face recognition laboratory presented during the course[1] was reused, but, the network was adapted in order to improve the performances with the dataset used for this project as show in Figure 4. Even if it doesn't satisfy completely the request, this part is an important and interesting part to investigate since it lets us to exploit a different technique not only based on a binary classification but based on a "multi-class" classification.

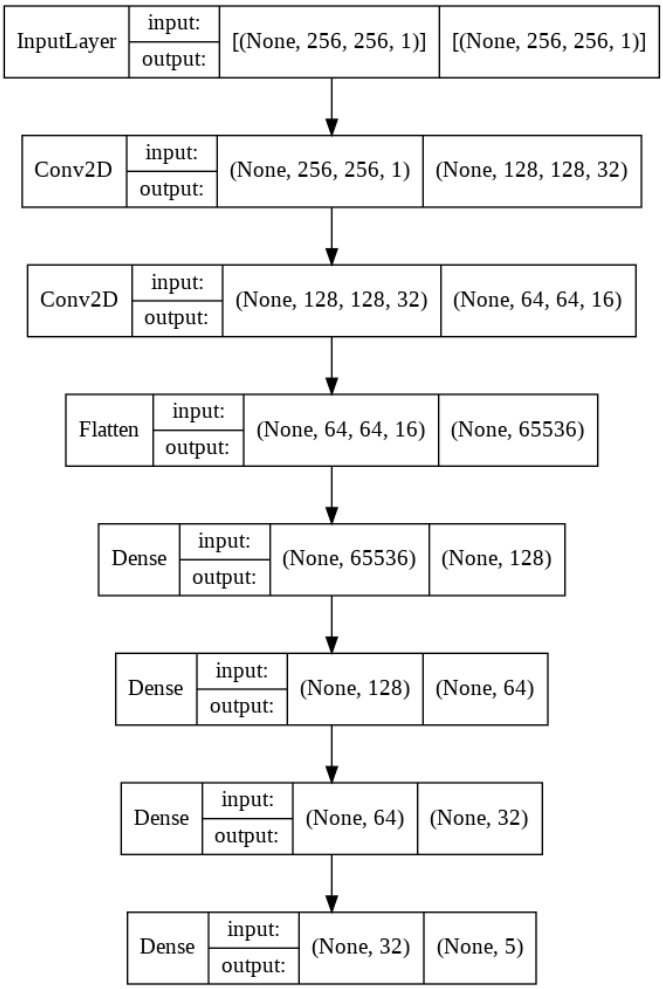

Figure 4: Design of the classification network

With this network and using as loss function the sparse categorical cross entropy loss, the accuracy is about 90.33% for the test set. In Figure 5 and Figure 6 an example of output and the confusion matrix respectively are show.

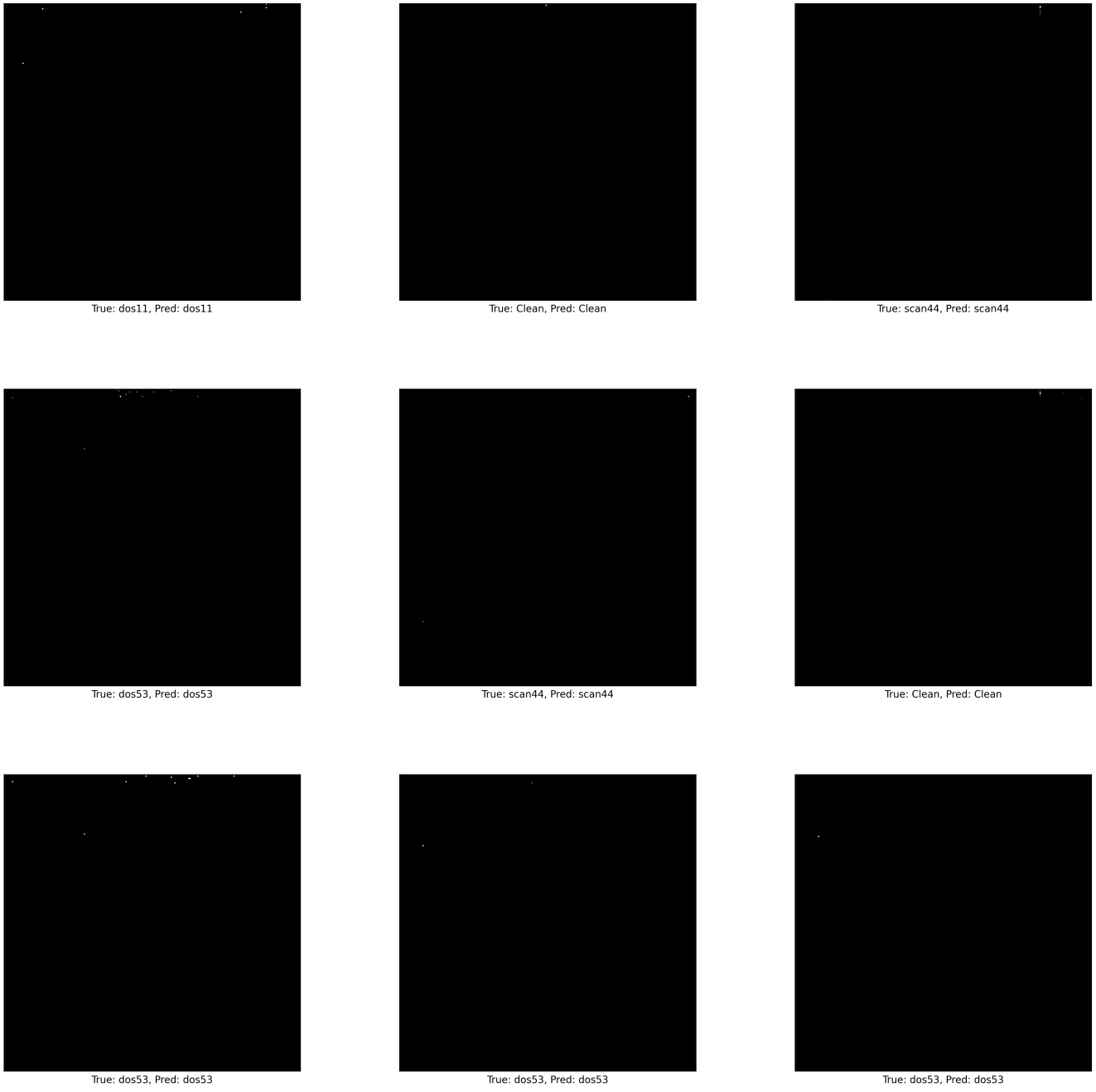

Figure 5: Example of output computed during the classification

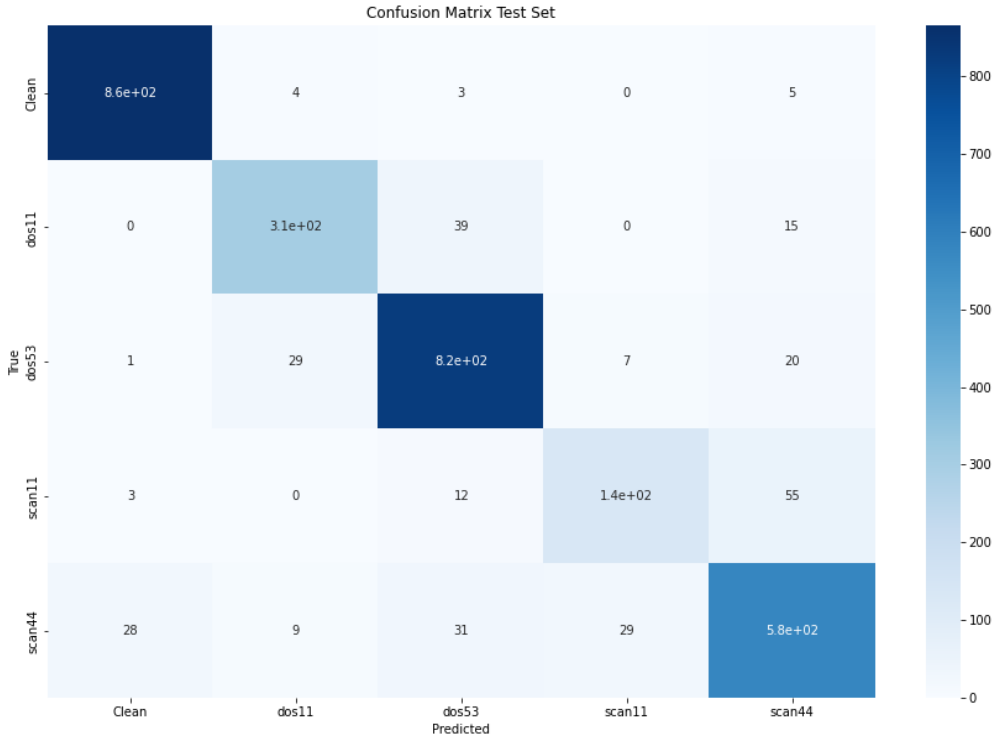

Figure 6: Confusion matrix of the test set

It is easy to see that, for each class, the performances are very good with an high number of correctly classified samples. The worst class is *scan44*, not only because it has the biggest number of not correctly classified samples, but also because, looking at at the dataset, the images from that class are very similar to the *clean* samples. This fact lead a lot of samples from *scan44* to be classified as *clean* (w.r.t. the other classes).

References

[1] Simone Milani, *Digital Forensics lectures*, AY 2021/2022, University of Padova, https://elearning.dei.unipd.it/course/view.php?id=8610

[2] https://ai.plainenglish.io/convolutional-autoencoders-cae-with-tensorflow-97e8d8859cbe

[3] https://www.analyticsvidhya.com/blog/2022/01/complete-guide-to-anomaly-detection-with-\\autoencoders-using-tensorflow/