Homework 3

Unsupervised learning: new sample generation

Martina Evangelisti 1796480

January 10, 2021

Contents

1 Introduction

The goal of this report is to describe a generative model which creates new samples similar to the ones taken from a given dataset.
The report will discuss two different solutions based on GANs, *Generative Adversarial Networks*. A GAN combines two networks: a generator and a discriminator. The first proposed solutionon is obtained modeling the discriminatior and the generator with convolutional neural network; the second one presents two neural networks which don't contain convolutional layers.

2 Generative Adversarial Network

As i said before, a GAN is composed by two main actors: a generator and a discriminator. The goal of the generator is to produce new data with the same distribution as the training set samples, while the discriminator has to identify if a sample comes from the distribution of the dataset or not. Practically it has to distinguish between the real samples coming from the dataset and the fake ones generated by the generator. On the contrary the generator aims at creating samples to be classified as real by the discriminator. This contraposition enables the model to learn in an unsupervised manner.

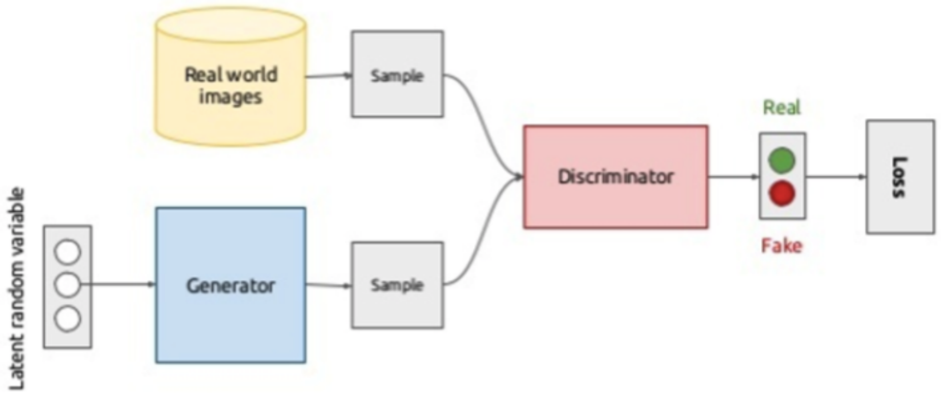
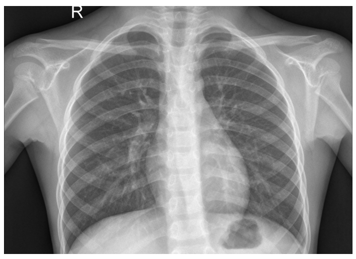


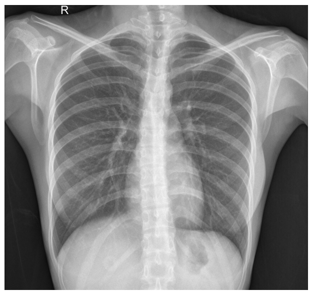
Figure 1: Structure of a GAN
[3]

3 Dataset

The main Dataset I used in this work is taken from Kaggle's site [4] and shows images of Chest X-Ray. The original dataset was split into 2 different classes: normal and pneumonia. The first one containing images of healty chests and the other one of chest having pneumonia.
From this Dataset i took only images from the class: normal.

Examples of images in the dataset:

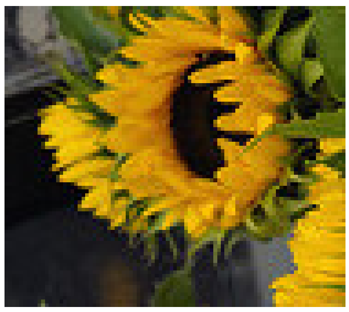




3.1 Additional Dataset

In order to show the versatility of GAN's application, i used the same model built for the previous dataset, in order to generate samples from a completely different dataset. This second is also taken from Kaggle [4] and contains images of different types of flowers. From this Dataset i took only images from the class: sunflower and i cropped the images around the shape of one sunflower.

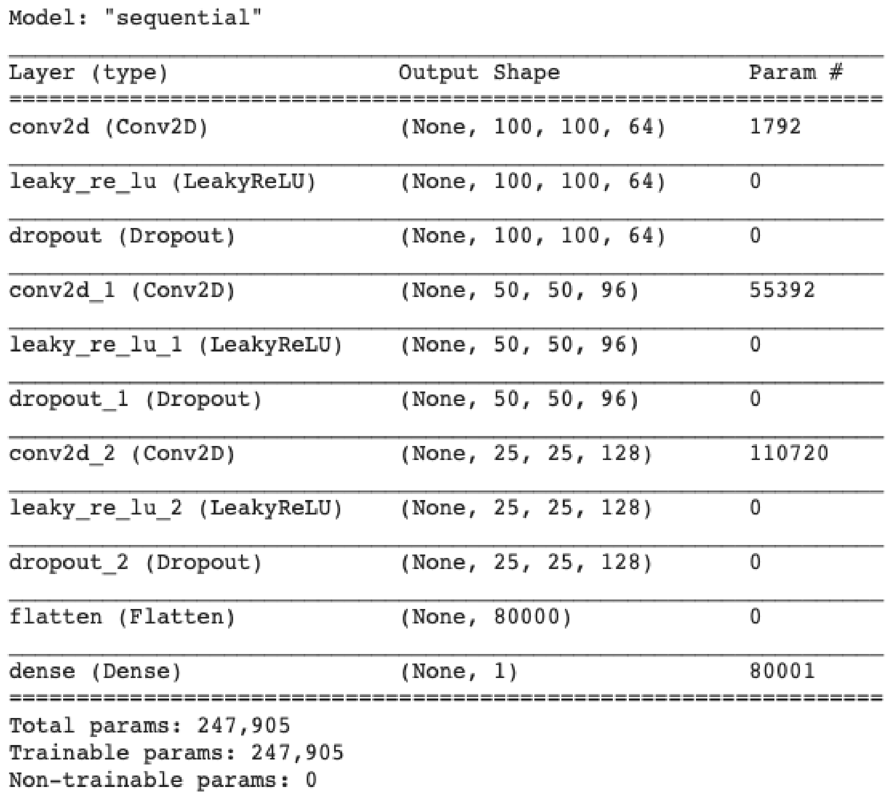
Examples of images in the dataset:



RX-Chest Dataset contains 1583 images while Sunflower Dataset contains 888 samples. With `ImageDataGenerator` i re-scaled all the images and i pre-processed the Dataset adding some **augmentation**: horizontal flip to randomly flip images and 10 degrees of rotation range for random rotations. Once instantiated the `ImageDataGenerator`, i loaded the images using `flow_from_directory` method.

4 GAN n.1: with CNN

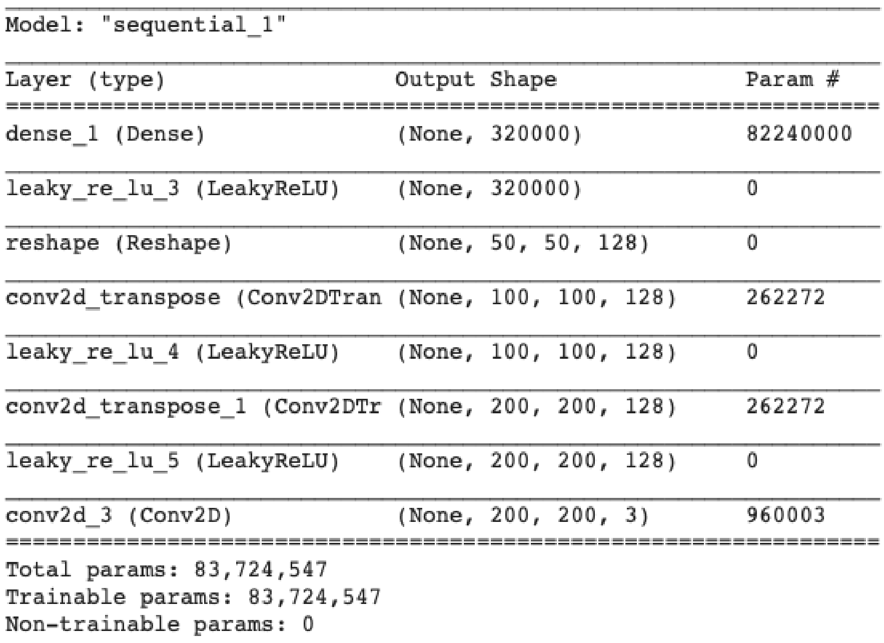
In the first GAN i designed both the generator and the discriminator are implemented with convolutional neural networks.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 100, 100, 64)      1792
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 100, 100, 64)      0
_________________________________________________________________
dropout (Dropout)            (None, 100, 100, 64)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 50, 50, 96)        55392
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 50, 50, 96)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 50, 50, 96)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 25, 25, 128)       110720
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 25, 25, 128)       0
_________________________________________________________________
dropout_2 (Dropout)          (None, 25, 25, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 80000)             0
_________________________________________________________________
dense (Dense)                (None, 1)                 80001
=================================================================
Total params: 247,905
Trainable params: 247,905
Non-trainable params: 0
```

Figure 2: Discriminator Layers

The Discriminator model has 11 layers: in particular 3 different convolutional layers each one followed by Leaky RELU activation function and a Dropout layer. At the end there is a flatten layer and a Dense layer with dimension equal to 1.
The Generator takes random noise as its input and has to transform it into a meaningful sample. The model is made of 8 layers. It starts from a foundation for 50x50 images and then with `Conv2DTranspose` and `LeakyRelu` it upsamples to 100x100 and then to 200x200.



```
_________________________________________________________________
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 320000)            82240000
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 320000)            0
_________________________________________________________________
reshape (Reshape)            (None, 50, 50, 128)       0
_________________________________________________________________
conv2d_transpose (Conv2DTran (None, 100, 100, 128)     262272
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 100, 100, 128)     0
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 200, 200, 128)     262272
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 200, 200, 128)     0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 200, 200, 3)       960003
=================================================================
Total params: 83,724,547
Trainable params: 83,724,547
Non-trainable params: 0
```

Figure 3: Generator Layers

The combined model is made by a concatenation of the Generator and the Discriminator, with the Discrimination layers set to non-trainable. It takes random noise as input and it has to generate new samples and determine their validity.

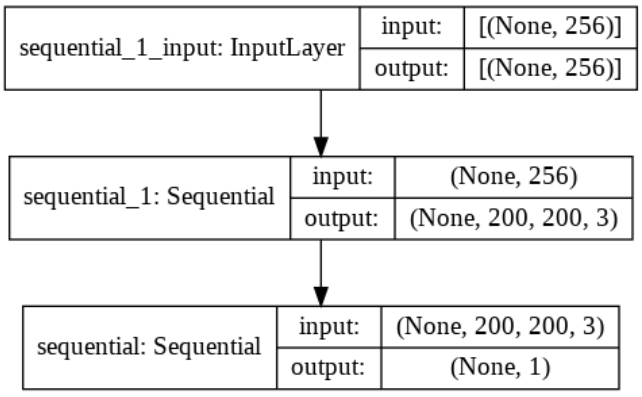


Figure 4: Combined Model

4.1 Training

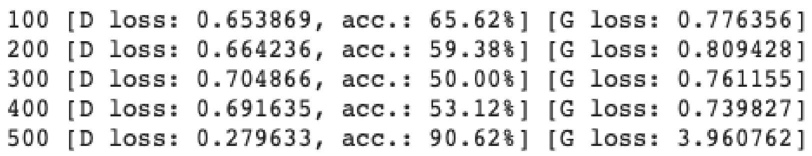
The Training phase can be split into 2 subphases:

1. Training the discriminator
2. Training the generator

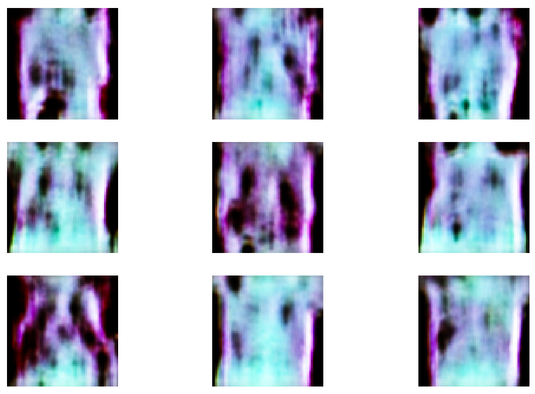
In the first phase the discriminator is trained with a set made of two parts with dimension equal to half of the batch size: one part contains images randomly taken from the Dataset and labeled with 1 (which means they're real), the other half is made of fake images created by the generator labeled with 0 (which stands for fake).

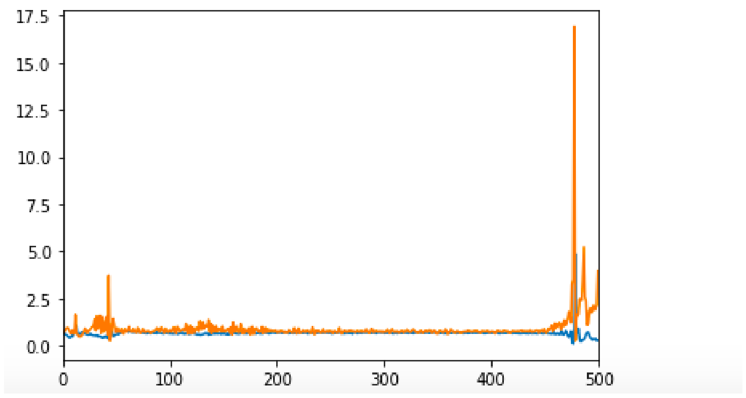
In the second phase the generator is trained in the combined model with noise labeled as real images cause it wants the discriminator to label the generated samples with 1.

These phases are repeated for each single epoch of the training.
I trained the model for 10000 epochs with a batch size of 32 and i saved and showed the updates every 100 epochs in order to show the losses of the discriminator and the generator. Every 500 epochs i showed some examples of images generated at that precise epoch in order to make the improvements visible.
Here there are parts of the training phase that shows this evolution:

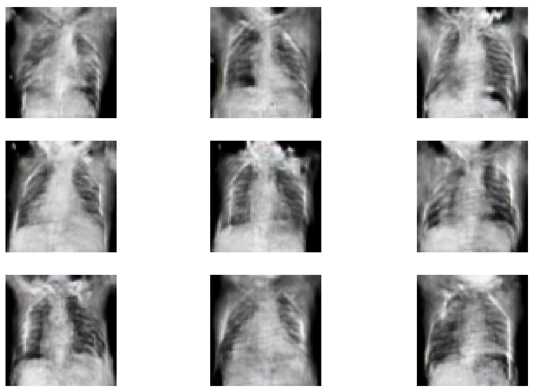


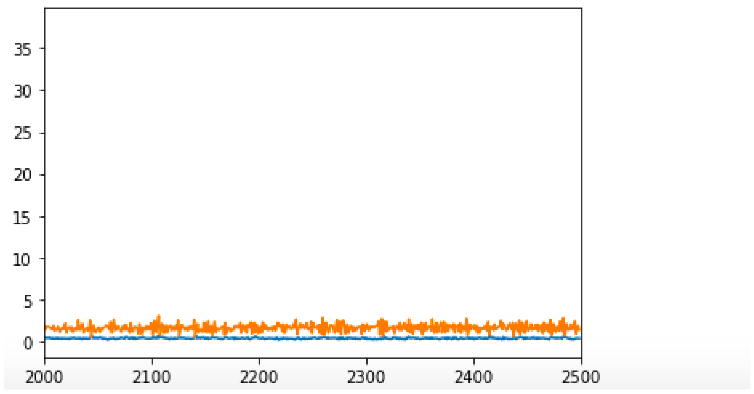
```
100 [D loss: 0.653869, acc.: 65.62%] [G loss: 0.776356]
200 [D loss: 0.664236, acc.: 59.38%] [G loss: 0.809428]
300 [D loss: 0.704866, acc.: 50.00%] [G loss: 0.761155]
400 [D loss: 0.691635, acc.: 53.12%] [G loss: 0.739827]
500 [D loss: 0.279633, acc.: 90.62%] [G loss: 3.960762]
```

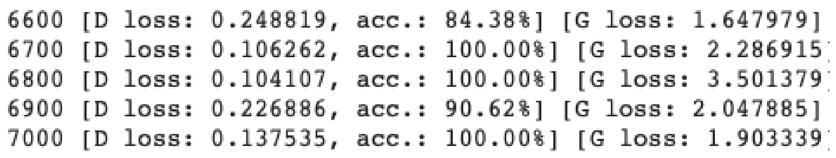




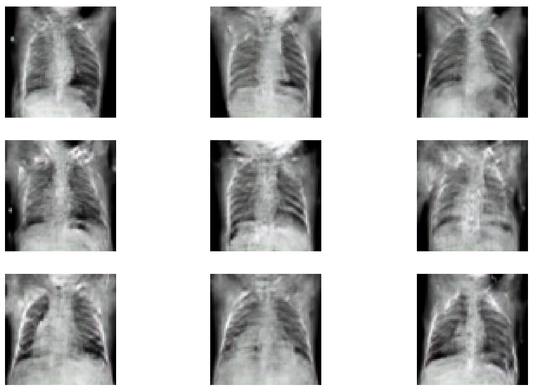
```
2100 [D loss: 0.364347, acc.: 81.25%] [G loss: 1.488375]
2200 [D loss: 0.379694, acc.: 84.38%] [G loss: 1.695908]
2300 [D loss: 0.301832, acc.: 93.75%] [G loss: 1.168600]
2400 [D loss: 0.452726, acc.: 87.50%] [G loss: 1.606034]
2500 [D loss: 0.351637, acc.: 84.38%] [G loss: 1.385527]
```

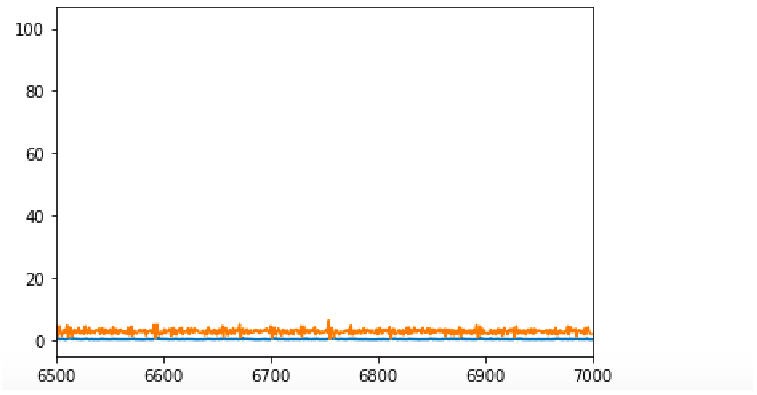






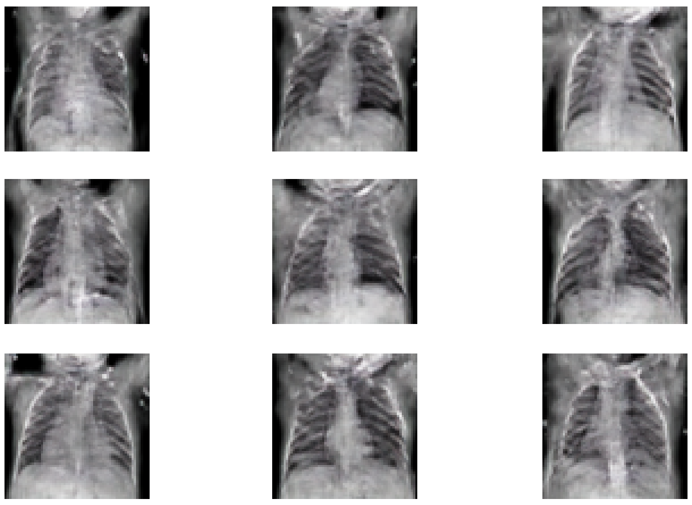
```
6600 [D loss: 0.248819, acc.: 84.38%] [G loss: 1.647979]
6700 [D loss: 0.106262, acc.: 100.00%] [G loss: 2.286915
6800 [D loss: 0.104107, acc.: 100.00%] [G loss: 3.501379
6900 [D loss: 0.226886, acc.: 90.62%] [G loss: 2.047885]
7000 [D loss: 0.137535, acc.: 100.00%] [G loss: 1.903339
```





4.2 Results

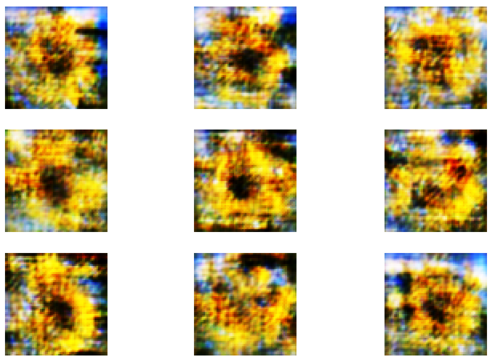
The generated images, despite the dimensions of the samples are smaller than the original images taken from the dataset are pretty similar to the original ones,
So, it is possible to identify them as representative samples of the class. Here there are some examples of generated images:



4.3 Training and results on the second Dataset

With the same model defined for the RX Chest Database, with different dimensions for the images, i trained the GAN in order to create samples of a completely different dataset composed by sunflower's images. Even in this case the training is made of 10000 epochs.

```
100 [D loss: 0.593339, acc.: 78.12%] [G loss: 0.939779]
200 [D loss: 0.687646, acc.: 56.25%] [G loss: 0.594424]
300 [D loss: 0.655217, acc.: 65.62%] [G loss: 0.734903]
400 [D loss: 0.671845, acc.: 62.50%] [G loss: 0.806798]
500 [D loss: 0.662684, acc.: 68.75%] [G loss: 0.793724]
```



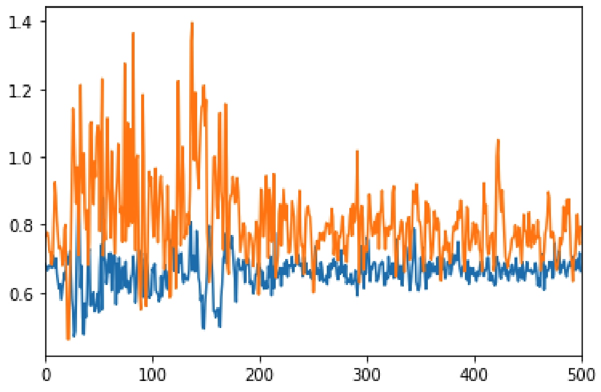


Figure 5: GAN after 500 epochs

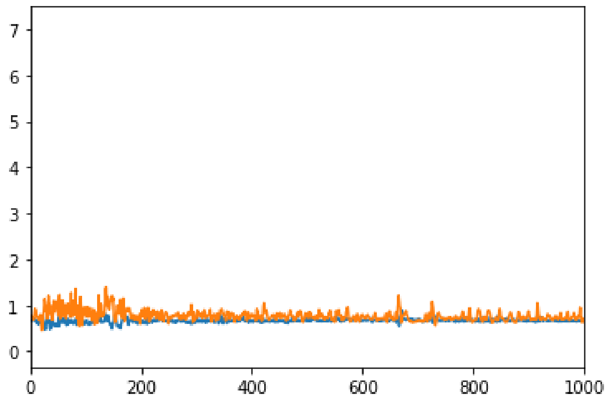
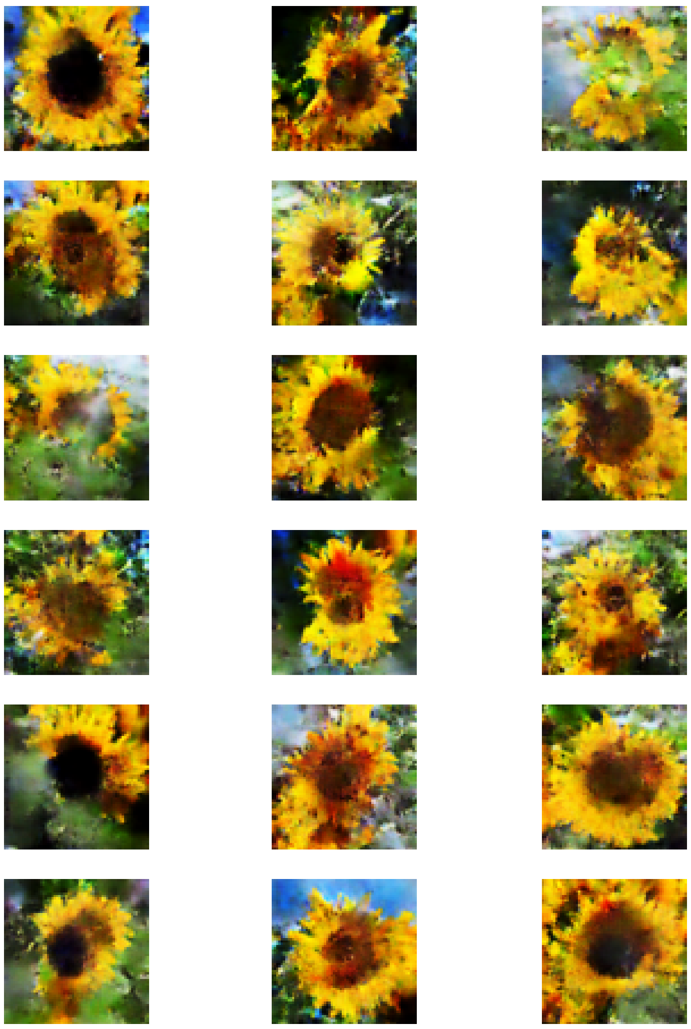


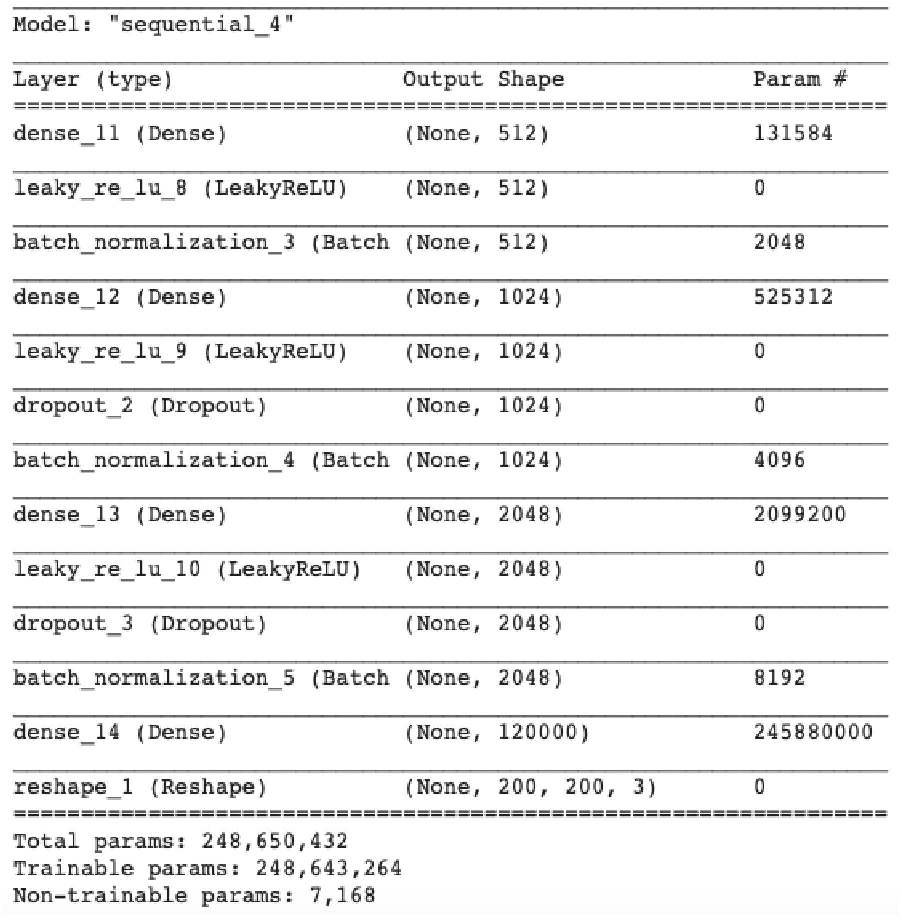
Figure 6: plot of the losses after 1000 epochs

As we can see from the plot we start from two opposite plots of the losses, in fact when the loss of the generator is high, the one of the discriminator is low and then they stabilize.
Even with this dataset that contained colored images the result are acceptable, even if the shape of the sunflowers are not always precise.
Here there are examples of the generated images.



5 GAN n.2

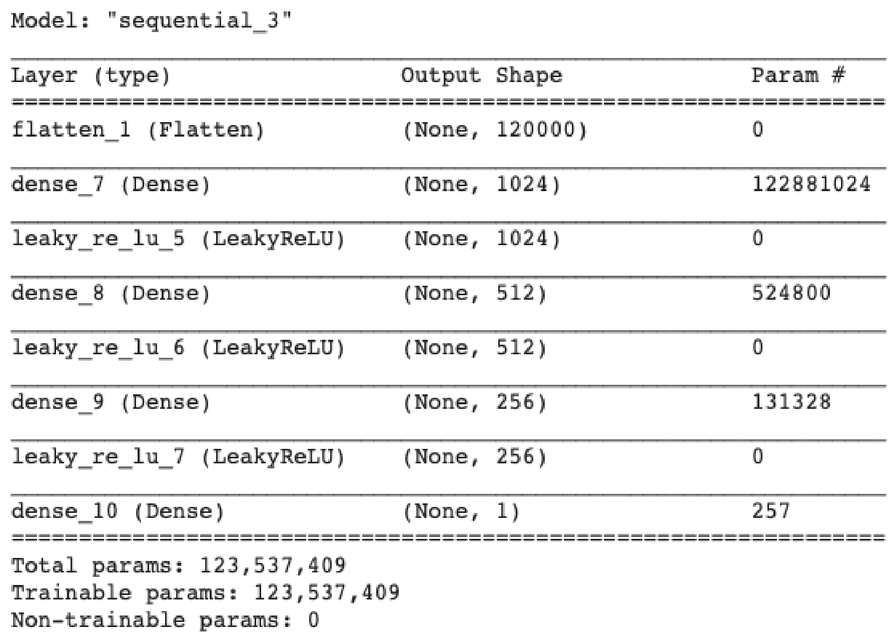
The second generative adversarial network i implemented is made of two neural networks, one for the generator and the other for the discriminator, which don't contain convolutional layers.



```
_________________________________________________________________
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_11 (Dense)             (None, 512)               131584    
_________________________________________________________________
leaky_re_lu_8 (LeakyReLU)    (None, 512)               0         
_________________________________________________________________
batch_normalization_3 (Batch (None, 512)               2048      
_________________________________________________________________
dense_12 (Dense)             (None, 1024)              525312    
_________________________________________________________________
leaky_re_lu_9 (LeakyReLU)    (None, 1024)              0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 1024)              0         
_________________________________________________________________
batch_normalization_4 (Batch (None, 1024)              4096      
_________________________________________________________________
dense_13 (Dense)             (None, 2048)              2099200   
_________________________________________________________________
leaky_re_lu_10 (LeakyReLU)   (None, 2048)              0         
_________________________________________________________________
dropout_3 (Dropout)          (None, 2048)              0         
_________________________________________________________________
batch_normalization_5 (Batch (None, 2048)              8192      
_________________________________________________________________
dense_14 (Dense)             (None, 120000)            245880000 
_________________________________________________________________
reshape_1 (Reshape)          (None, 200, 200, 3)       0         
=================================================================
Total params: 248,650,432
Trainable params: 248,643,264
Non-trainable params: 7,168
```

Figure 7: Generator Layers

The Generator is made of 13 layers: we have 4 Dense Layers always followed by leakyRELU and batch_normalization with some dropouts. Last layer applies a reshape to the dimensions of the sample images.



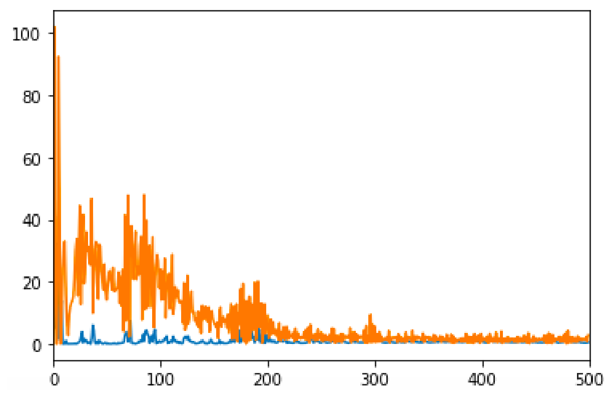
```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 120000)            0
_________________________________________________________________
dense_7 (Dense)              (None, 1024)              122881024
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 1024)              0
_________________________________________________________________
dense_8 (Dense)              (None, 512)               524800
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 512)               0
_________________________________________________________________
dense_9 (Dense)              (None, 256)               131328
_________________________________________________________________
leaky_re_lu_7 (LeakyReLU)    (None, 256)               0
_________________________________________________________________
dense_10 (Dense)             (None, 1)                 257
=================================================================
Total params: 123,537,409
Trainable params: 123,537,409
Non-trainable params: 0
```

Figure 8: Discriminator Layers

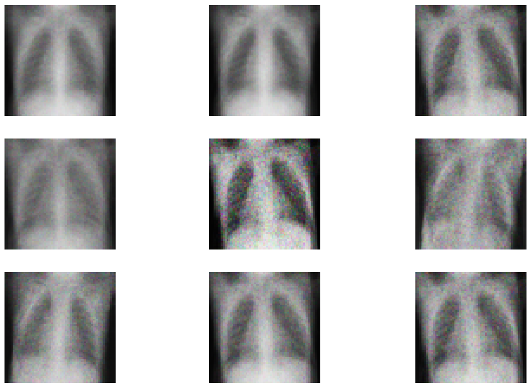
The Discriminator model has 8 layers: the first one is a Flatten layers all the others are an alternation of Dense and LeakyRELU with one last Dense layer with dimension equal to 1.

5.1 Training

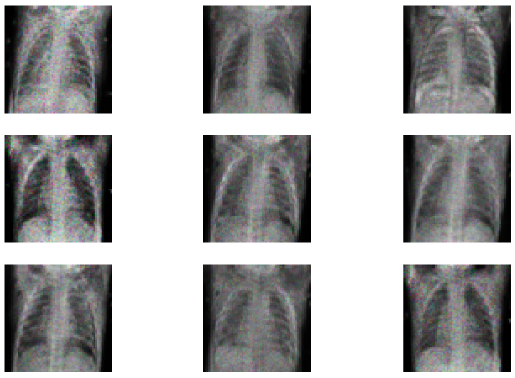
The GAN is trained in the same way as done for the model explained before. It has been trained for 10000 epochs with a batch size of 32. Here there are parts of the training phase that shows this evolution:

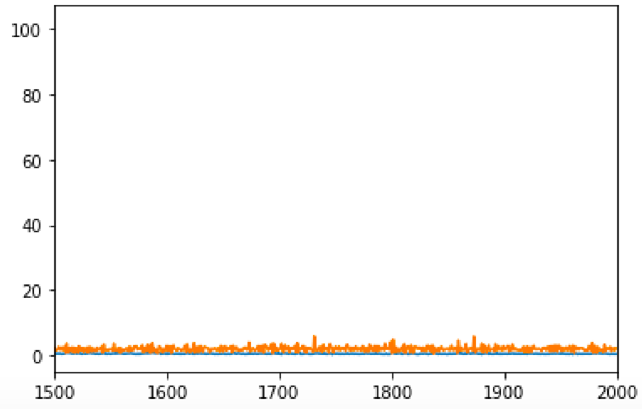


```
100 [D loss: 0.947219, acc.: 90.62%] [G loss: 17.995493]
200 [D loss: 2.493396, acc.: 56.25%] [G loss: 6.427276]
300 [D loss: 2.010633, acc.: 50.00%] [G loss: 0.648027]
400 [D loss: 1.055534, acc.: 50.00%] [G loss: 3.307745]
500 [D loss: 0.500393, acc.: 68.75%] [G loss: 2.880703]
```

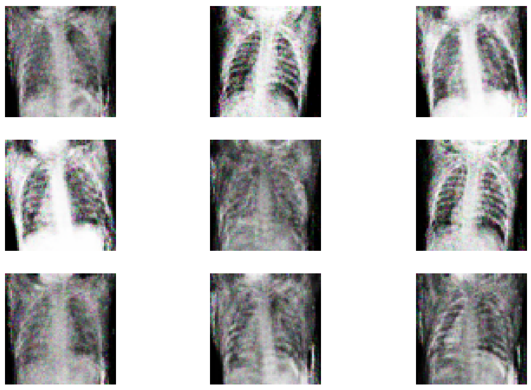


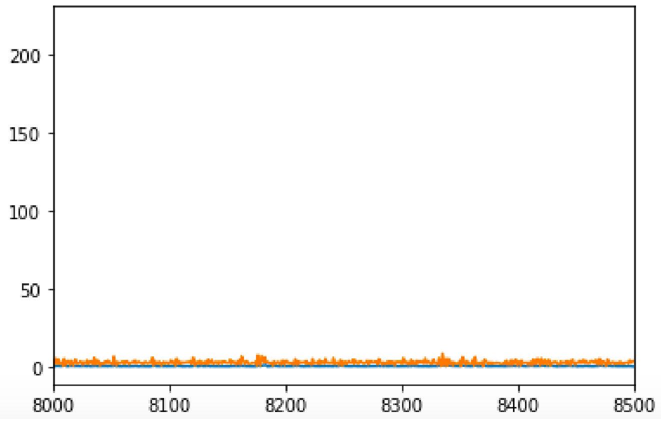
```
1600 [D loss: 0.432508, acc.: 78.12%] [G loss: 1.818657]
1700 [D loss: 0.621153, acc.: 68.75%] [G loss: 1.364106]
1800 [D loss: 0.466021, acc.: 71.88%] [G loss: 4.084167]
1900 [D loss: 0.335336, acc.: 84.38%] [G loss: 2.791069]
2000 [D loss: 0.300615, acc.: 96.88%] [G loss: 2.069652]
```





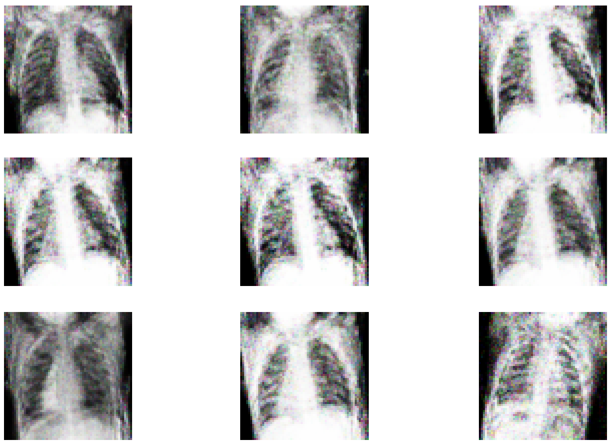
```
8100 [D loss: 0.322194, acc.: 84.38%] [G loss: 1.716190]
8200 [D loss: 0.307054, acc.: 81.25%] [G loss: 1.123918]
8300 [D loss: 0.104508, acc.: 100.00%] [G loss: 2.985170]
8400 [D loss: 0.602482, acc.: 75.00%] [G loss: 0.965219]
8500 [D loss: 0.096140, acc.: 100.00%] [G loss: 3.706502]
```





5.2 Results

From the images obtained from this GAN it is possible to recognize the structure of an RX Chest, but the images are less precise with respect to the ones obtained from the model previously described. Here are some examples of generated samples:



6 Conclusions

This report presented two different implementation of a Generative Adversarial Network to generate new samples similar to the images belonging to the original dataset. This method proved to be of fundamental importance and obtained pretty good result, on the other hand the training of a GAN is particularly intensive. Even in this case it's shown that Convolutional neural networks are powerful tools to deal with images and the results shown previously obtained with convolutional layers are better than the ones obtained without them.

References

[1] Tensorflow and Keras documentation
`https://www.tensorflow.org/api_docs/python/tf/all_symbols`

[2] Keras documentation
`https://keras.io/api/`

[3] Slide 15. Dimensionality Reduction // L. Iocchi, F.Patrizi

[4] Kaggle
RX Chest Dataset
`https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia`
Flower Dataset
`https://www.kaggle.com/joeylimzy/flowers`