UNIVERSITÀ DI PISA

MSc in Artificial Intelligence and Data Engineering
Deep Learning
A.A. 2020/21

Convolutional Neural Network for Medical Imaging Analysis

Abnormality detection in mammography

Project Report

Matteo Dessì
Guido Gagliardi

Contents

1 Introduction

The aim of the project is to resolve, by using deep neural networks, two breast classification problems:

- Benign-Malign Classification
- Mass-Calcification Classification

The dataset typically used is CBIS-DDSM, Curated Breast Imaging Subset of Digital Database for Screening Mammography.

The dataset is composed of high resolution images in DICOM format. In the original collection, for each image numerous information are stored, including the binary mask used to locate the abnormality.

Since the DICOM format is not natively supported by *tf.keras*, the dataset has been treated and converted into the equivalent dataset provided to us, containing *NumPy* arrays for images and labels of both training and testing set.

First we analyzed the dataset,visualizing images, studying the image composition and the distribution of the labels.

Then, we searched for numerous papers on academic portals, both at the beginning of the project writing in order to identify commonly used approaches and eventual difficulties that we might have encountered, and during the development in order to address specific problems or find new solutions necessary to improve the adopted methods.

To solve the requested classification problems we built networks from scratch and used pre-trained networks. Afterwards, the structure previously used for the networks from scratch has been re-used to build a Siamese Architecture specifically for the mass-calcification problem. Finally, an Ensemble of the models previously saved for the benign-malign problem has been created to obtain better performance. Many of the functions used in the pre-processing phase are common between all the networks, therefore we have decided to group them all together into a single file, the 'utility.ipynb'.

2 Papers

Before starting to work on the problem, we have read numerous works carried out on the DDSM data set. By reading the papers, we learned of several useful techniques later applied on our work.

First we find a general approach to solve this problem that has been presented in the papers [2][8] to understand the main issues and steps to achieve the solutions.

The list of the papers is in the bibliography.

To address the Pretrained network task, we found out which are the most popular networks used in the documentations [1] [3] [7]:

- VGG16
- InceptionV3
- ResNet50
- AlexNet

For our project we decided to adopt the VGG16 and InceptionV3 networks.

In the paper [6] the authors decided to add a new SVM classifier on top of the last convolution layer of the pretrained network, that is therefore used as a features extractor for the fully connected layer installed on top of it. Having a small dataset, we decided to adopt this technique too, but deciding to use an MLP classifier instead of the SVM classifier used in the paper. The level at which we cut the network to extract the features has been considered as an hyper parameter.

Regarding the pre-processing, the numerous papers we have found had to address the problem of extracting the ROI, Region Of Interest, patch before giving it as input to the network, a problem we didn't have to face in our dataset. By reading the subsequent remaining steps of preprocessing, we decided to use an intersting approach: adopt either the CLAHE algorithm or another function to enhance the contrast, and where needed or aimed at obtaining more "structural" images, adopt filter to reduce noise or extract specific frequencies.

To face overfitting we used ourselves the most adopt technique of the papers [6], namely data augumentation, constisted in applying horizontal flip and rotations. Furthermore in some cases the system has also need to perform a drop out step in order to avoid the overfitting problem.

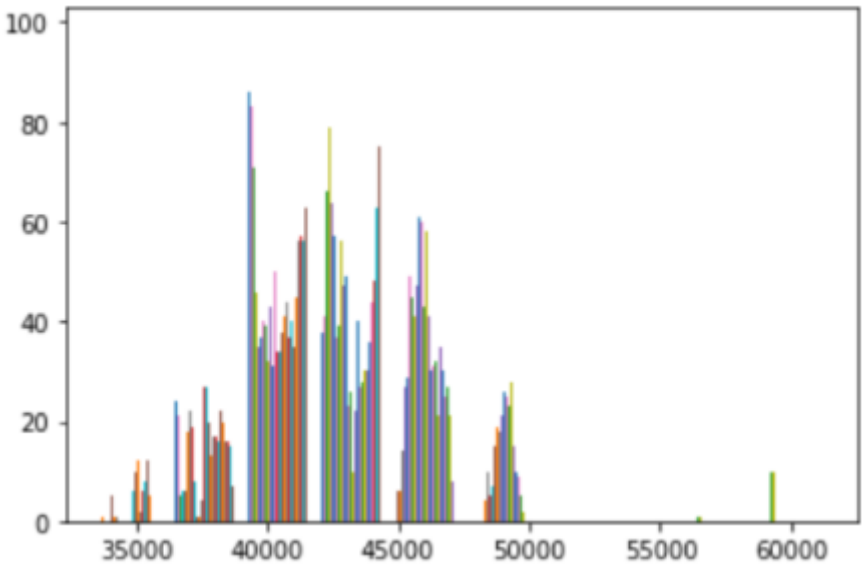
Finally, in order to address the unbalance of the classes in our problem, we decided to adopt as suggested in the papers [9][4] the approach of creating a GAN, Generative Adversarial Network,in order to create syntethic samples of the minority classes.

3 Data Visualization and Preprocessing

As previously described, the dataset provided for this project is different from the original CBIS-DDSM, since some pre-processing to extract the abnormality patches was already performed. The dataset is provided in the form of *NumPy* arrays. The *NumPy* images provided are 16-bit gray-scale images, meaning that there is a single channel for the images which pixels values range from 0 to 65536.

Histogram representative of the pixel values of a sample image:

Figure 1: Image values



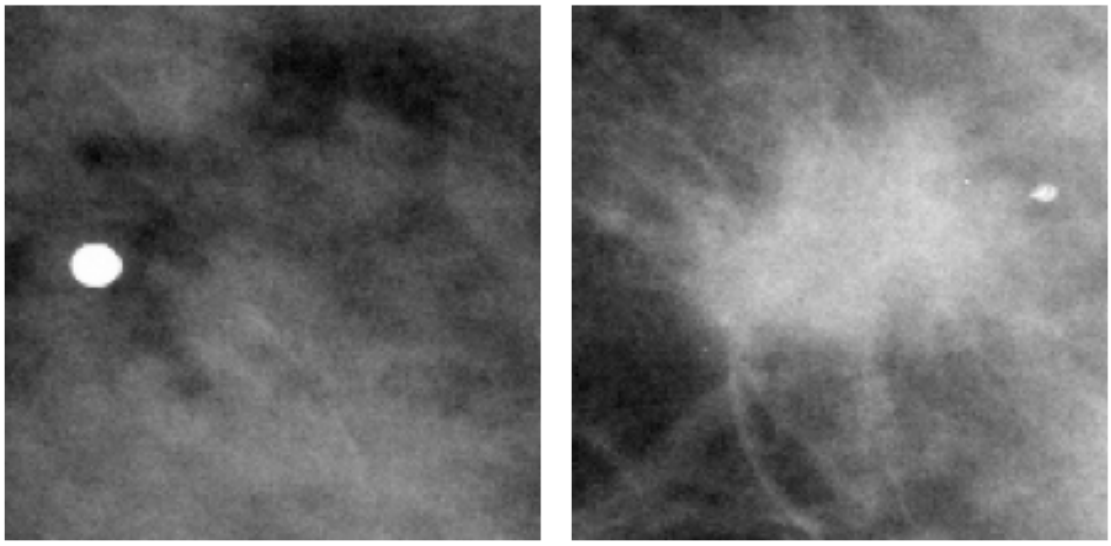
For each patient the dataset gives both a Baseline (healthy tissue) patch and an Abnormality patch. The Baseline patches are located in the even position of the images arrays while the corresponding Abnormality patches are in the consequent odd indexes.

The label are mapped in the following manner:

- 0: Baseline patch
- 1: Mass, Benign
- 2: Mass, Malignant
- 3: Calcification, Benign
- 4: Calcification, Malignant

Example of Baseline patch and corresponding Abnormality patch:

Figure 2: Pair example

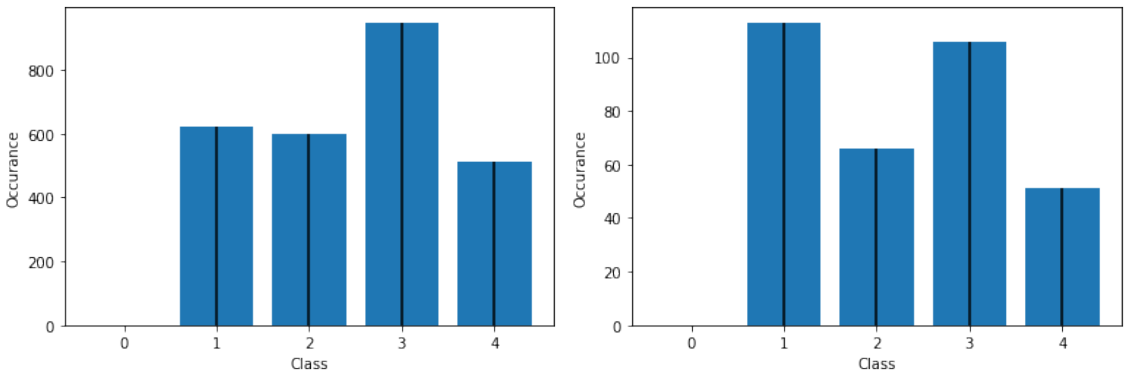


(a) Baseline first sample

(b) Abnormal first sample

The dataset label distribution of the classes is the following:

Figure 3: Data set distribution



(a) Training set

(b) Testing set

For each task we constructed derivative dataset starting from the original one while also modifying the original labels depending on the type of task to be executed. We had to build three different dataset:

- Benign vs Malignant dataset
- Mass vs Calcification dataset
- Siamese dataset

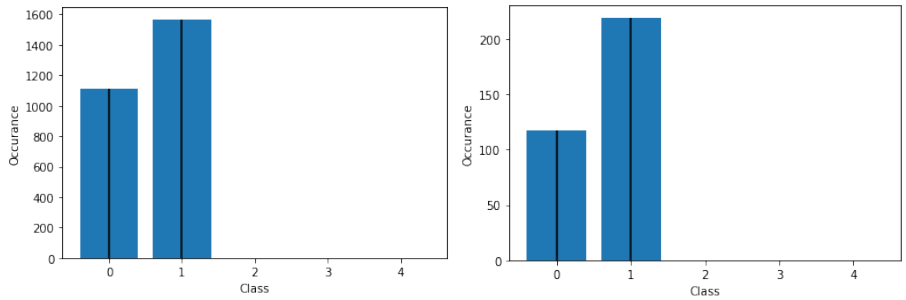
3.1 Data set creation

3.1.1 Benign vs Malignant data set creation

In the Benign vs Malignant classification task we selected only the odd indexes from both the images and labels *NumPy* arrays to retrieve all the Abnormalities and their labels. In addiction we performed a translation of the original labels in order to conform them to the task, in the following manner:

- $OriginalLabels\ 2, 4 \rightarrow Label\ 0$
- $OriginalLabels\ 1, 3 \rightarrow Label\ 1$

Figure 4: Benign vs Malignant data set



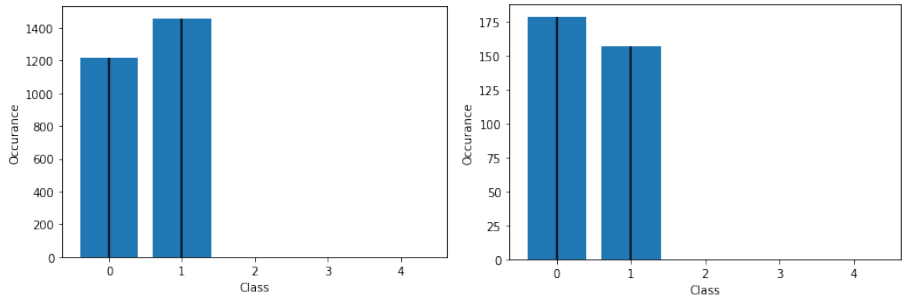
(a) Training set (b) Testing set

3.1.2 Mass vs Calcification data set creation

In the Mass vs Classification classification task we selected only the odd indexes from both the images and labels *NumPy* arrays to retrieve all the Abnormalities and their labels. In addiction we performed a translation of the original labels in order to conform them to the task, in the following manner:

- $OriginalLabels\ 1, 2 \rightarrow Label\ 0$
- $OriginalLabels\ 3, 4 \rightarrow Label\ 1$

Figure 5: Mass vs Calcification data set



(a) Training set (b) Testing set

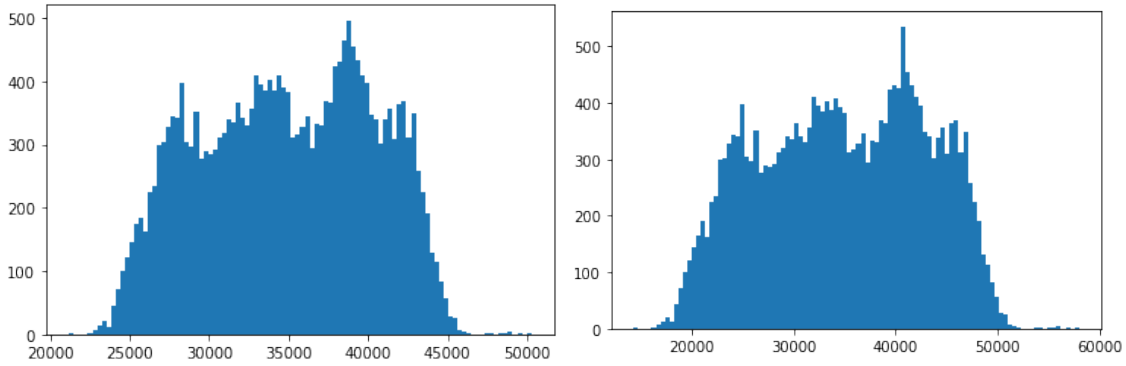
3.2 Preprocessing

3.2.1 Enhance Images

Through reading the paper [6] we found out that a useful techniques to improve the image quality for classification tasks consists in increasing the contrast between the white and black pixels composing the images, stressing the general structure of the image itself. For this reason we compared two different approaches, one proposed by us, and one suggested by the authors of the article. The first is a standard way to enhance the contrast of the images in tensorflow, the second one instead belongs to a class of adaptive contrast enhancement (AHE) described by [10].

Enhance Contrast A first test was performed by applying the *tensorflow* adjust _contrast function to both training and testing images. We can see in the next pictures how the pixel values of the first abnormality patch were modified by the function:

Figure 6: Frequencies contrast comparison



(a) Original frequencies

(b) Contrast frequencies

And a visual representation of effect of the application:

CLAHE CLAHE(Contrast Limited AHE) is a variant of adaptive histogram equalization in which the equalization is performed in small patches or in small tiles with high accuracy and contrast limiting.

The second test was conducted by applying the *cv2*function createClahe using standard parameters to both the training and testing set. In the next pictures we can clearly see how the application modified the pixel value distribution of the first abnormality:

And a visual representation of the effect:

The CLAHE algorithm applied to the images has the undesired effect of producing a large amount of noises as described in [6], therefore we thought about applying some kind of filters.

Figure 7: Contrast image comparison

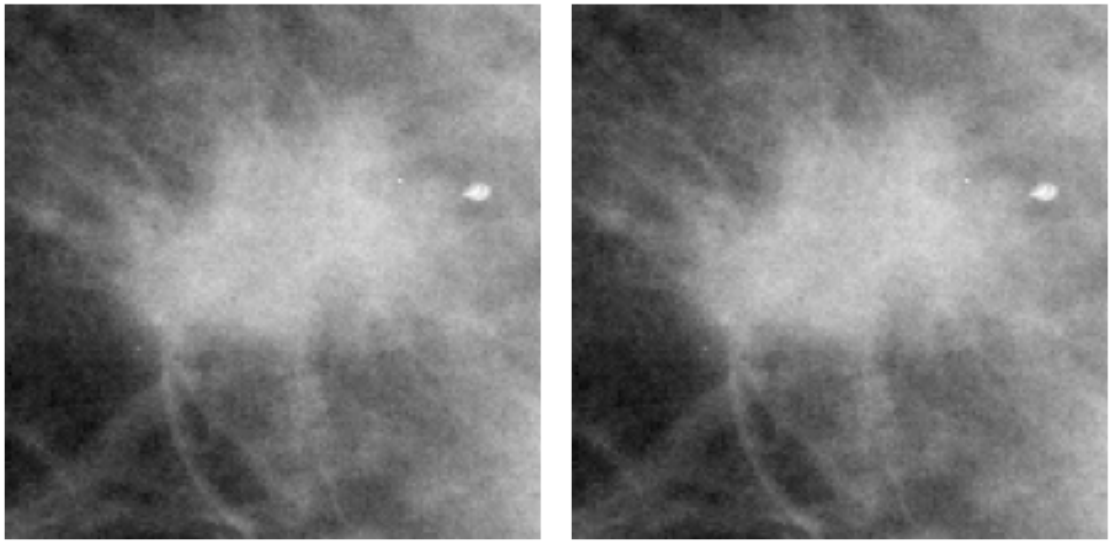
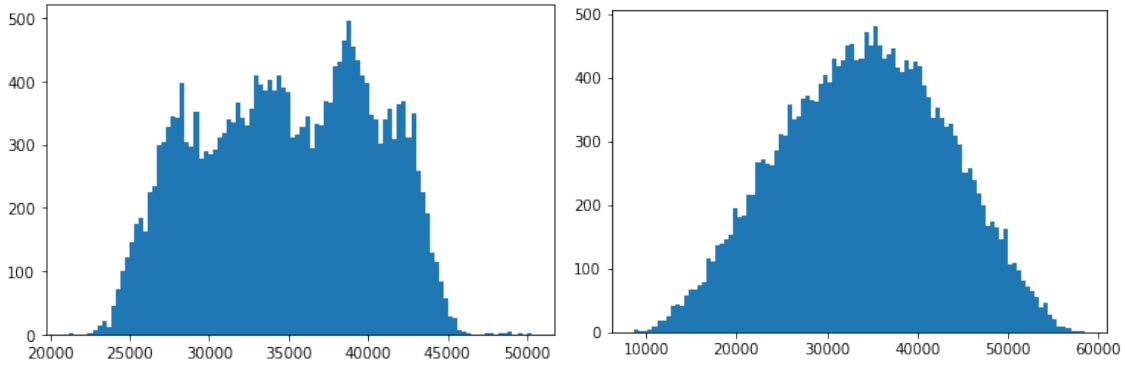


Figure 8: Frequencies CLAHE comparison



(a) Original frequencies

(b) CLAHE frequencies

3.2.2 Filters

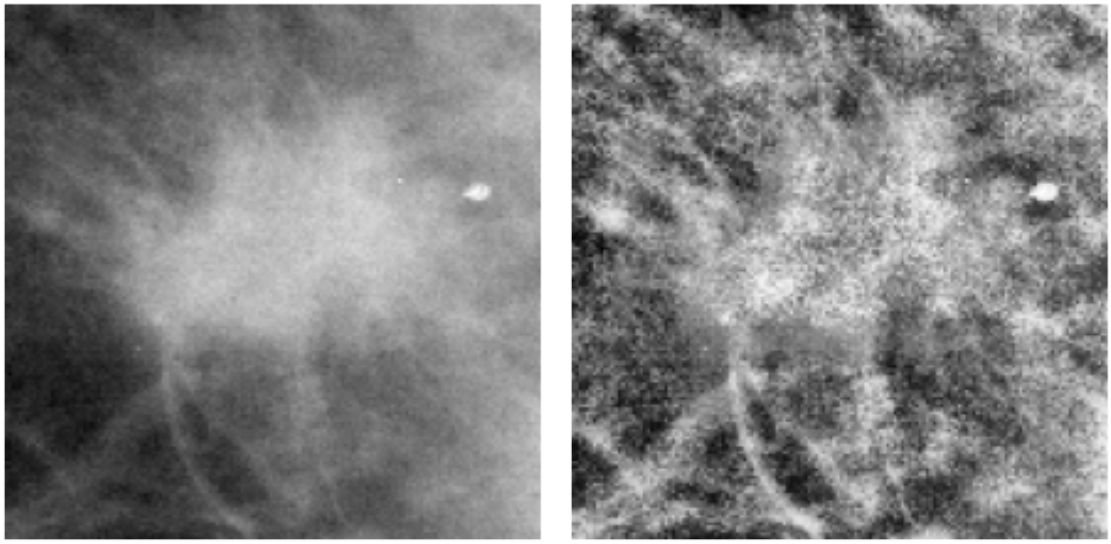
Through previous studies, we learned the usefulness of filters for:

- Smoothing a signal: Decresing the noise and eventual outliers
- Selecting specific frequencies: in our case we used this functionality in order to isolate the frequencies (namely, the pixel image values) that contain the general information of the structure of the abnormalities,in order to help the CNN to better distinguish the different shape for the classification task

In order to face these issues three approaches has been studied: *Threshold*, *High Pass* and *Savitzky Golay*.

Threshold The first type of filter we tried to apply is the thresholding method.

Figure 9: CLAHE image comparison



The thresholding methods replace each pixel in an image with a black pixel if the image intensity is less than some fixed constant T(the threshold) or a white pixel if the image intensity is greater than that constant.

In order to apply such technique, we called the *cv2* library function 'treshold', that takes as argument three parameters:

1. thresholdvalue: the value which is used to classify the pixel values
2. maxValue: the value to be given if the pixel value is more (or less) than the threshold
3. thresholdTechnique: the type of thresholding applied

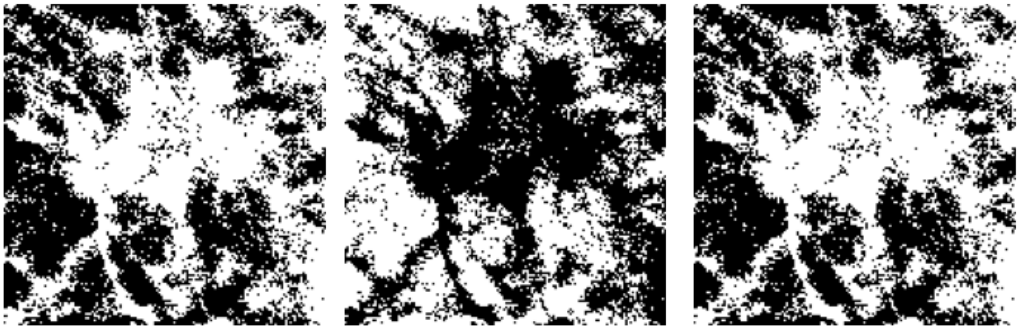
As suggested in the official *cv2* documentation we studied during our researches, we decided to inspect three different type of thresholdTechnique:

- cv2.THRESH _BINARY: If the pixel intensity is greater than the set threshold, the value is set to 255, else to 0(black)
- cv2.TRESH _BINARY _INV: Inverted or opposite case of cv2.TRESH _BINARY
- cv2. THRESH _BINARY + cv2.THRESH _OTSU: Automatically set the threshold to the optimal one using OTSU.

The first two technique requires to find an optimal value, usually located in the middle of the histogram of the image values. Such process (performed manually in the case of 10a 10b) however can be generalized by using OTSU , as shown in 10c.

After testing these approaches on several images and establishing the goodness of the last one, that solution has been adopted for the following work.

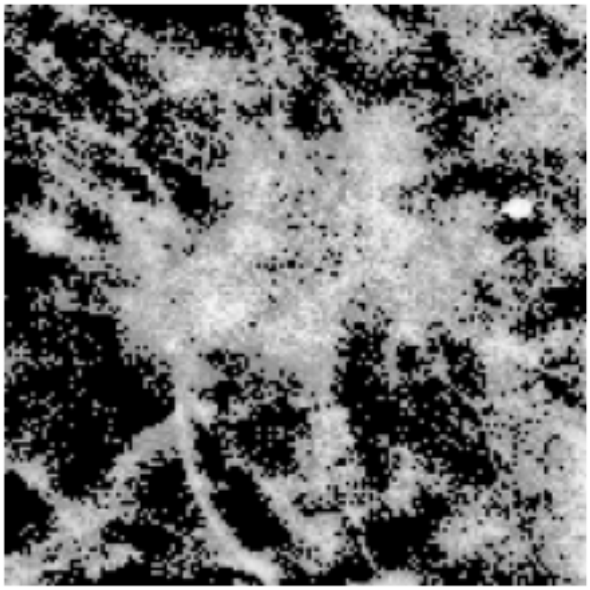
Figure 10: Tresholding techniques



(a) Binary (b) Binary Inv (c) Binary+Otsu

High Pass The High Pass filter we applied conserved all the superior half of the image pixel values while putting the rest of the pixel to 0. An example of the result is the following:

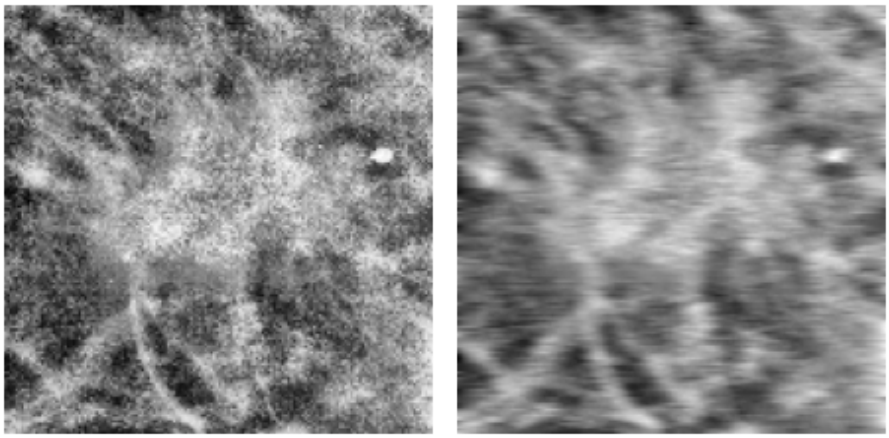
Figure 11: High Pass filter



Savitzky Golay The Savitzky Golay filter is a digital filter that replaces each pixel value with a linear combination of itself and a certain number (defined by the window) of its neighbours. We both tried to apply a small window, which results in very little modification of the original image, and a wide window, causing an unwanted distortion of the image.

After several tests we decide to implement our filter phase using the *thresholding method with binary + Otsu technique*, the one which performed better in term of highlighting the abnormalities.

Figure 12: Savitzky Golay



(a) Small window (b) Big window

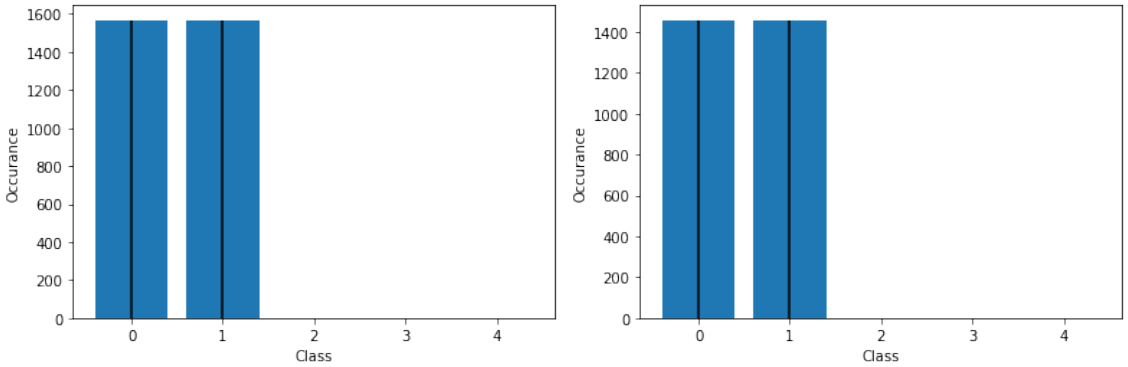
3.2.3 Normalization

Before passing the images in input to the CNNs, a normalization was applied to the image pixels transforming their values from [0,65536] to [0,1].

3.3 Class Balancing

As we can notice in the 3.1, both the Benign vs Malignant and Mass vs Calcification training sets are unbalanced, a problem stressed in particular in the Benign vs Malignant training set in which the difference of tuples between the majority and minority class is equal to 460, while in the Mass vs Calcification training set the difference is "only" of 240. The approach we decided to adopt is the 'oversampling' in which we increase the amount of tuples belonging to the minority class until the count is equal to the one of the majority class. What we expect to obtain is depicted in the following images:

Figure 13: Oversample Distribution



(a) Benign vs Malignant (b) Mass vs Calcification

3.3.1 Random Sampling with replacement

The first type of oversampling we decided to adopt is the random sampling from the minority class with replacement.

In other words, we will randomly select tuple from the minority class and add it to the training set, basically creating a copy of an existing tuple.

The advantage of this technique consists in the simplicity of the implementation, while the big disadvantage consist in the replication of the images itself: by replicating such a big number of tuple of such small samples pool, we will cause a future undesired overfitting of the models on the replicated tuples,a problem heavily important for the Benign vs Malignant training set because the copies consist of, more or less, a third of the entire minority class. Furthermore the replication process doesn't expand the input space, providing the network with new information. For this reason it will be more difficult for the network to understand the implicit structure of the data.

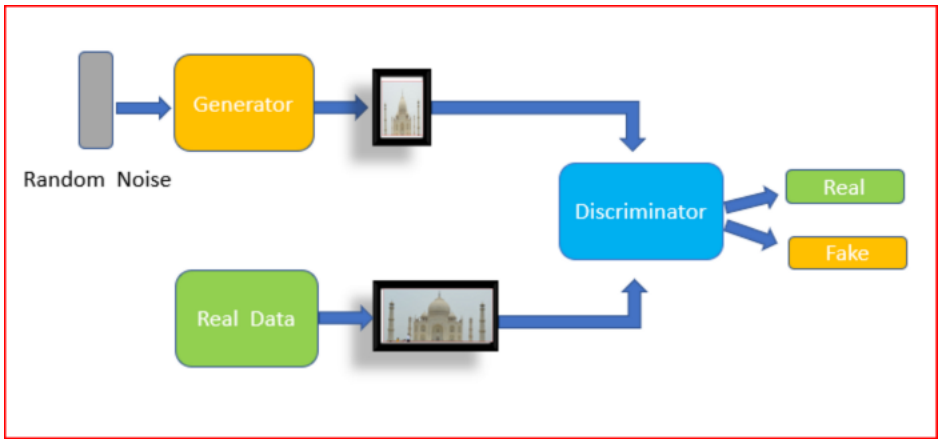
The two ways to solve this problem are:

- Acquire new real data to extend the dataset.
- Try to synthesize new artificial data.

Since the first solution was impractical in the contest of this work, we implemented the second one utilizing a Variational Auto-encoders approach, using Generative Adversarial Network.

3.3.2 Generative Adversarial Network

Figure 14: GAN



GAN is an unsupervised deep learning algorithm where we have a Generator pitted against an Adversarial network called Discriminator. The Generator's objective will be to generate data that is very similar to the training data. Data generated from Generator should be indistinguishable from the real data. The Discriminator takes two sets input,one input comes from the training dataset(real data) and the other input is the dataset generated by Generator. Its aim is to distinguish real from false data. In our case, Generator and Discriminator are both neural network that run in competition with each other in the training phase. The steps are repeated several times and in this, the Generator and Discriminator get better and better in their respective jobs after each repetition.

Figure 15: GAN Structure

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 256)	25856
leaky_re_lu (LeakyReLU)	(None, 256)	0
dense_1 (Dense)	(None, 512)	131584
leaky_re_lu_1 (LeakyReLU)	(None, 512)	0
dense_2 (Dense)	(None, 1024)	525312
leaky_re_lu_2 (LeakyReLU)	(None, 1024)	0
dense_3 (Dense)	(None, 22500)	23062500

Total params: 23,745,252
Trainable params: 23,745,252
Non-trainable params: 0

(a) Generator Structure

Model: "sequential_1"

Layer (type)	Output Shape	Param #
dense_4 (Dense)	(None, 1024)	23041024
leaky_re_lu_3 (LeakyReLU)	(None, 1024)	0
dropout (Dropout)	(None, 1024)	0
dense_5 (Dense)	(None, 512)	524800
leaky_re_lu_4 (LeakyReLU)	(None, 512)	0
dropout_1 (Dropout)	(None, 512)	0
dense_6 (Dense)	(None, 256)	131328
leaky_re_lu_5 (LeakyReLU)	(None, 256)	0
dense_7 (Dense)	(None, 1)	257

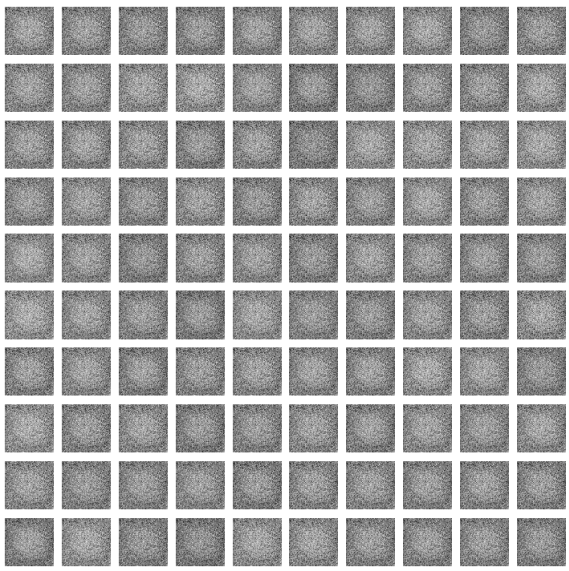
Total params: 23,697,409
Trainable params: 23,697,409
Non-trainable params: 0

(b) Discriminator Structure

The Generator will start at the first epoch by producing a random noise,that will model over time in order to emulate, at the best it can, the real data, which is in our case the minority classes of the Benignant vs Malign and Mass vs Calcification training sets.

As described in [9] and [4] the standard approach to generate artificial images from the

Figure 16: Initial Noise

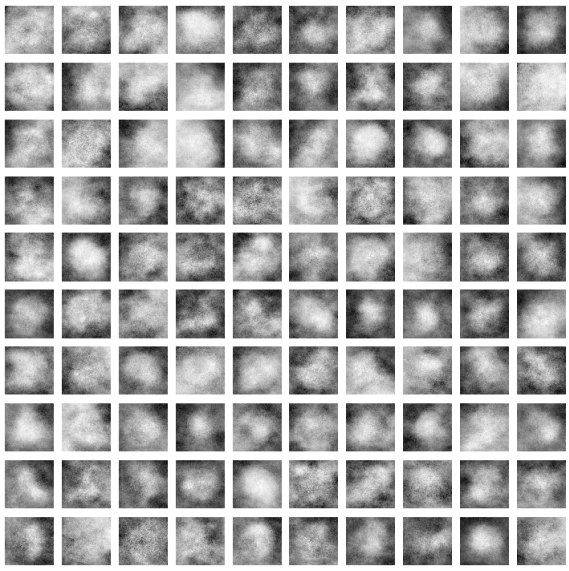


minority class should be done passing directly the ROI (in our case the images) to the GAN. In our experiments this method resulted to be very ineffective: the quality of the generated images was not comparable to the original ones.

For this reason we tried to simplify the work of the GAN, enhancing the images first and then passing them to network.

First we apply the simple*tensorflow* adjust contrast. However, because of the complexity of our images, we still weren't able to obtain satisfying result,not even after 1000 epochs, as the following image represents:

Figure 17: GAN on enhanced images, Mass vs Calcification, 1000 epochs



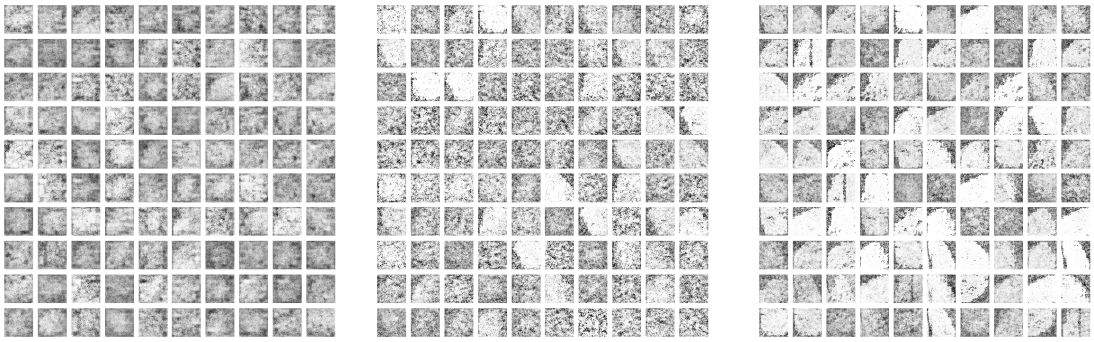
We then decided to apply a stronger form of contrast that we were able to obtain, as presented before, through the CLAHE, and then a thresholding filter with the thresholding technique cv2. THRESH _BINARY + cv2.THRESH _OTSU, that stresses heavily the

hidden structure of the abnormality images.

In the following images is depicted the progression through the epochs of the learning of the two GANs, one for each of the for minority class of the case study.

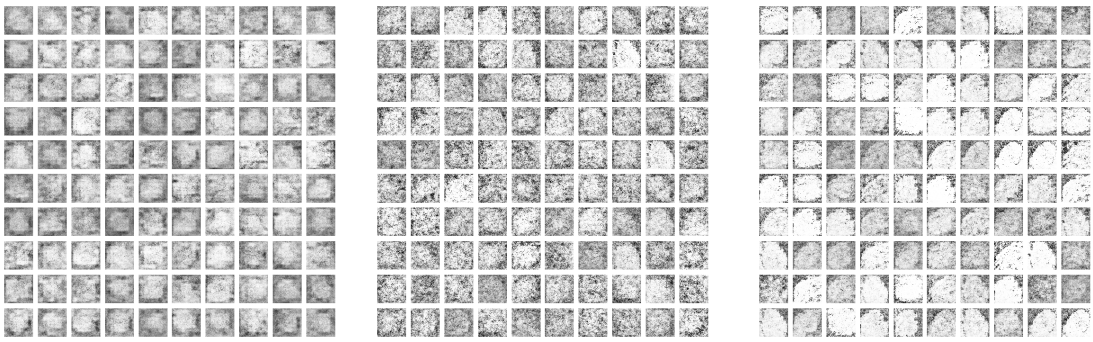
The neural networks are correctly able, at the end, to create synthetic abnormality patches.

Figure 18: GAN on Benignant vs Malignant, label 0



(a) 50 epochs (b) 300 epochs (c) 550 epochs

Figure 19: GAN on Mass vs Calcification, label 0



(a) 50 epochs (b) 300 epochs (c) 550 epochs

It's important to notice that since the oversampling phase processed in this session produces enhanced and filtered images, the whole dataset will have to be treated according to this procedure, in order to provide the CNNs with coherent and homogeneous data.

3.4 Dataset shuffling

By analyzing the dataset we noticed the fact that the images and relative labels are inserted into the *Numpy* arrays into blocks. In order to avoid such net division, we decided to randomly shuffle the dataset.

3.5 Dataset splitting

Having constructed the training set and test set, we proceed to obtain the validation set. In order to do so we used a function of the *sklear* library, setting the splitting factor to 0.2.

4 CNN from Scratch

4.1 Mass vs Calcification

The network used in the Mass-Calcification has the following architecture:

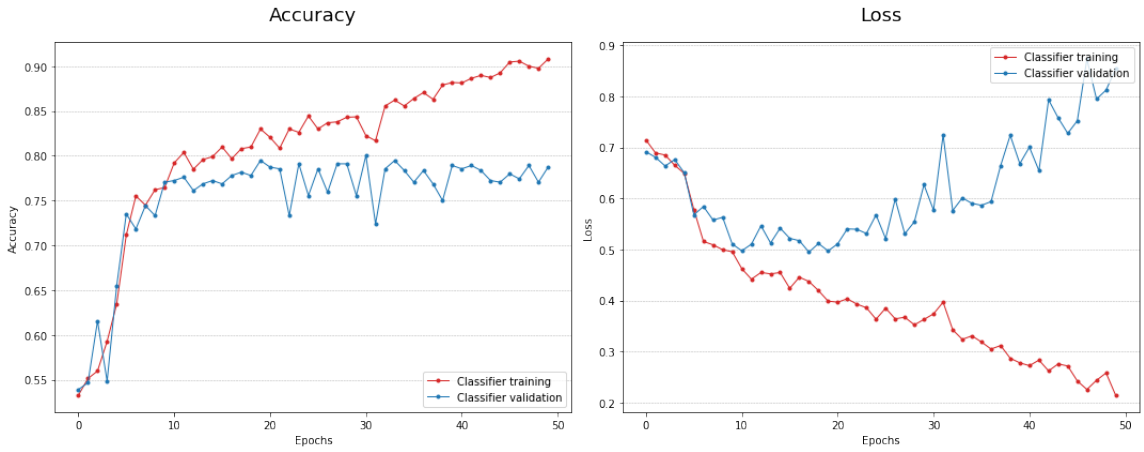
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 32)      320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 34, 34, 128)       73856
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 32, 32, 128)       147584
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 32768)             0
_________________________________________________________________
dense (Dense)                (None, 512)               16777728
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 17,018,497
Trainable params: 17,018,497
Non-trainable params: 0
_________________________________________________________________
```

In the preprocessing phase, the images has been normalized in a range from 0 to 1.

Since the class were unbalanced, we tried oversampling with both random sampling with replacement and syntetic mammography abnormalities created through the appropriate GAN.

In addition, we tried to enhance the contrast both through the *tensorflow* adjust _contrast and CLAHE algorithm, to which we made follow a tresholding filter.

The first results were not so good, we faced a severe overfitting:



This overfitting can be observed both from the images and in the difference in accuracy and loss between training and validation set:

	Training Set	Validation Set
Accuracy	0.91	0.79
Loss	0.21	0.85

To overcome this problem, we used Data Augmentation, as suggested in the literature, in particular by applying:

- horizontal flipping
- 90° rotation

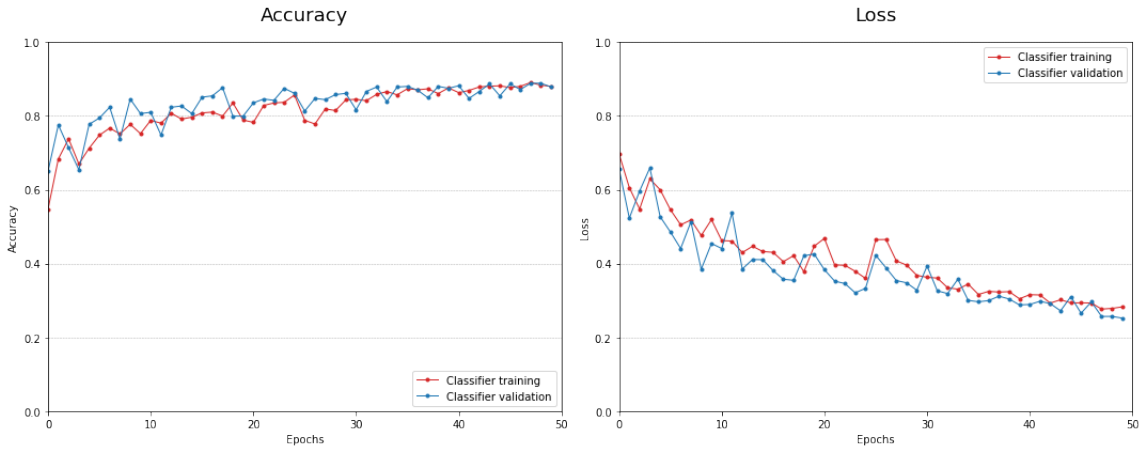
Other than Data Augmentation,we tried to obtain better results through the tuning of the hyper-parameters, specifically:

- learning rate
- optimizers
- number of epochs

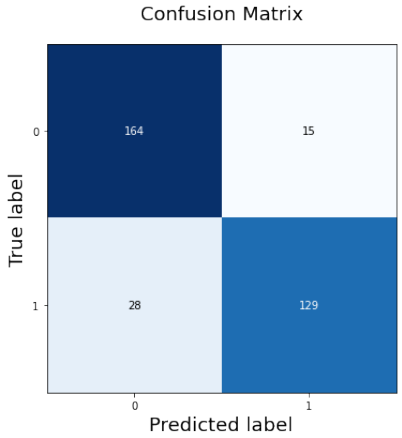
Following we have a table with the most important tests:

	Data Augumentation	Random Oversampling	GAN	Image Contrast	CLAHE	Filter	Optimize	Learning Rate	Epochs	Accuracy	Loss	AUC
Scratch_1							Adam	0,001	50	0.76	0.82	0.76
Scratch_2	✓						Adam	0,001	50	0.83	0.40	0.83
Scratch_3	✓	✓					Adam	0,001	50	0.83	0.38	0.83
Scratch_4	✓	✓		✓			Adam	0,001	50	0.87	0.29	0.84
Scratch_5	✓	✓		✓			RMSprop	0,001	35	0.78	0.47	0.79
Scratch_6	✓	✓		✓			Adam	0,003	35	0.85	0.38	0.85
Scratch_7	✓	✓			✓		Adam	0,001	50	0.79	0.48	0.79
Scratch_8	✓	✓			✓	✓	Adam	0,001	50	0.81	0.44	0.81
Scratch_9	✓		✓		✓	✓	Adam	0,001	30	0.82	0.41	0.82

The best model obtained is SCRATCH _4: the model will progressively improve both in term of accuracy and loss before stabilizing towards the final phase. This was verified in several tests.



The confusion matrix reflects the goodness of the results:



4.2 Benign vs Malignant

The network used in the Mass-Calcification has the same architecture of the Bening-Malignant one:

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 148, 148, 32)	320
max_pooling2d (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_1 (Conv2D)	(None, 72, 72, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 36, 36, 64)	0
conv2d_2 (Conv2D)	(None, 34, 34, 128)	73856
conv2d_3 (Conv2D)	(None, 32, 32, 128)	147584
max_pooling2d_2 (MaxPooling2	(None, 16, 16, 128)	0
flatten (Flatten)	(None, 32768)	0
dense (Dense)	(None, 512)	16777728
dense_1 (Dense)	(None, 1)	513

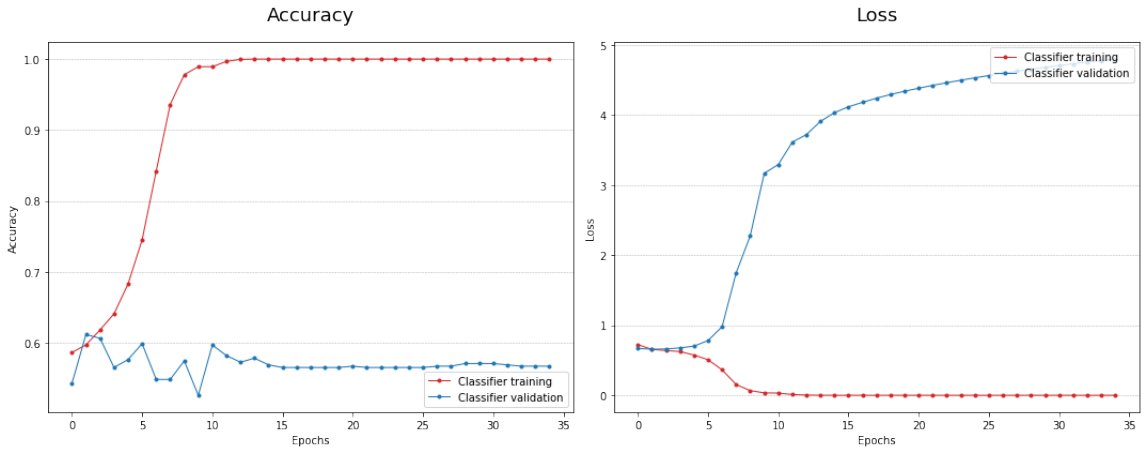
Total params: 17,018,497
Trainable params: 17,018,497
Non-trainable params: 0

In the preprocessing phase, the images has been normalized in a range from 0 to 1.

Since the class were unbalanced, we tried oversampling with both random sampling with replacement and syntetic mammography abnormalities created through the appropriate GAN.

In addition, we tried to enhance the contrast both through the *tensorflow* adjust _contrast and CLAHE algorithm,to which we made follow a tresholding filter.

The first results were not so good, we faced overfitting:



Overfitting that can be observed both from the images and in the difference in accuracy and loss between training and validation set:

	Training Set	Validation Set
Accuracy	0.91	0.64
Loss	0.2	1.55

To overcome this problem, we used Data Augmentation, as suggested in the literature, in particular by applying:

- 90° rotation
- horizontal flipping

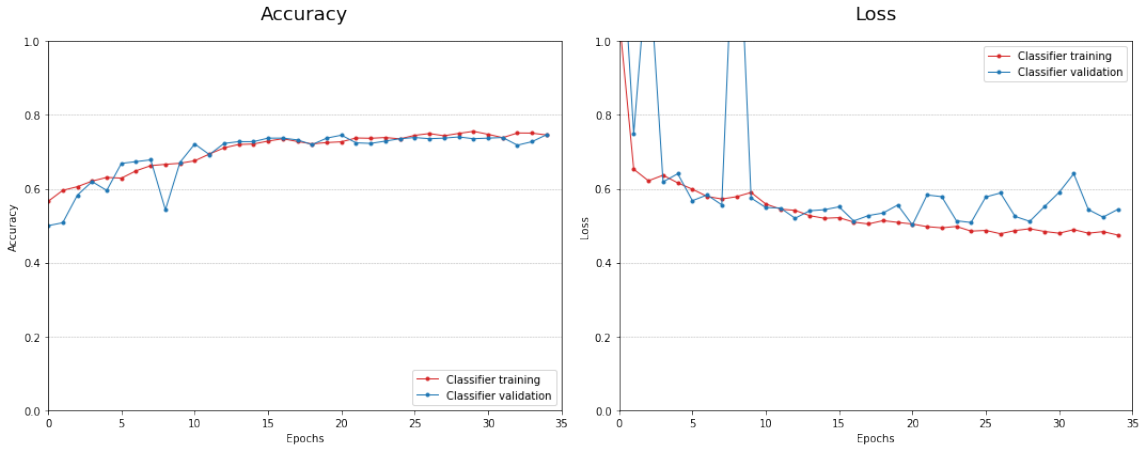
And a level of Dropout.
Other than Data Augmentation,we tried to obtain better results through the tuning of the hyper-parameters, specifically:

- optimizer
- number of epochs
- learning rate

Following we have a table with the most important tests:

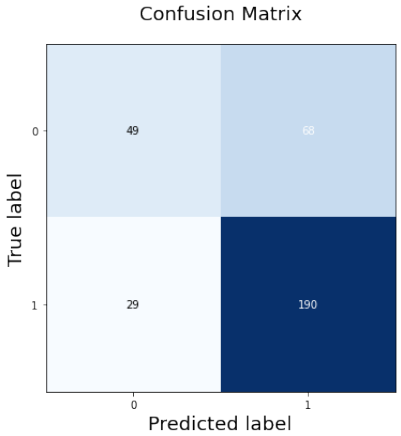
	Data Augmentation	Random Oversampling	GAN	Image Contrast	CLAHE	Filter	Optimiz	Learnin Rate	Epochs	Accurac	Loss	AUC	f2
Scratch_1							Adam	0,001	35	0.60	0.72	0.60	0.65
Scratch_2	✓						Adam	0,001	35	0.65	0.63	0.55	0.71
Scratch_3	✓	✓					Adam	0,001	35	0.61	0.64	0.61	0.61
Scratch_4	✓	✓		✓			Adam	0,001	35	0.64	0.62	0.61	0.70
Scratch_5	✓	✓			✓	✓	Adam	0,001	35	0.67	0.63	0.63	0.75
Scratch_6	✓		✓		✓	✓	Adam	0,001	35	0.70	0.63	0.61	0.81
Scratch_7	✓		✓		✓	✓	RMSprop	0,001	35	0.71	0.62	0.64	0.84

To evaluate our models we decided to consider primarily the accuracy and the f2 metric. The best model obtained is SCRATCH _6.



We have chosen as the best model the one with the best trade-off between accuracy and f2.

The results obtained,though not exceptional, are considerably better than the initial tries. The confusion matrix well represents the final performance:



5 Pretrained CNN

Following the papers we have studied, and the lessons of our pratical laboratories, we decided to adopt the following pretrained CNNs:

- VGG16
- InceptionV3

The pretrained networks, after removing one or more of the fully connected layers on top of them, have been used as features extractors for a MLP classifier added on top of the net.

5.1 Pre-Processing

In the pre-processing phase, the images has been normalized in a range from 0 to 1.

As observed before, we tried oversampling both using Random Oversampling with Replacement or the GANs.

The images have been treated either with the contrast enhancement or the CLAHE application plus a tresholding filter, a pair we have found as the the most successfull in the previous step.

As we have already observed in the GAN section 3.3.2, these syntetizers provide images already enhanced and filtered; since the fact that our pretrained networks are not trained on this kind of preprocessed ROI, it is reasonable to think that this approach will not be the more appropriate for this task.

5.2 From 1-channel to 3-channel images

The pre-trained architecture need to work with 3-channel images, but our dataset is composed of only grayscale images(and therefore a single channel).

A possibility was to just replicate the image on 3 channels before providing it to the pretrained models. The solution we decided to adopt instead consists in using a convolutional layer before the pretrained network to enlarge the depth from 1 to 3; in order to do this, the convolutional layer must have 3 filters of size 1x1.

5.3 Mass vs Calcification

5.3.1 VGG16

The comprehensive architecture used in this task, including the VGG16 pretrained network, include the layers of data augumentation, the convolutional layer to increase the depth of the input and the layers to classify the features.

Figure 20: Mass vs Calcification VGG16 Architecture

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
random_flip (RandomFlip)     (None, 150, 150, 1)       0
_________________________________________________________________
random_rotation (RandomRotat (None, 150, 150, 1)       0
_________________________________________________________________
conv2d (Conv2D)              (None, 150, 150, 3)       6
_________________________________________________________________
vgg16 (Functional)           (None, 4, 4, 512)         14714688
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0
_________________________________________________________________
dense (Dense)                (None, 512)               4194816
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 18,910,023
Trainable params: 4,195,335
Non-trainable params: 14,714,688
_________________________________________________________________
```

The hyper-parameters on which we performed tries where:

- the level to cut
- the optimizer
- the learning rate
- the batch size

We tried to cut the pretrained at different levels in order to find out if the representation extracted on the upper levels could be more useful for the classification task.

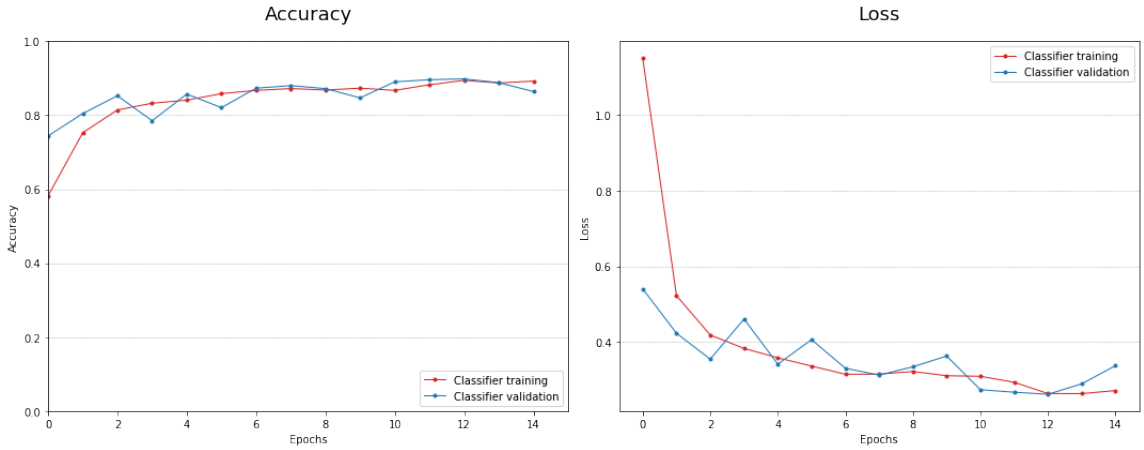
Here are a results of the better results:

	Data Augumentation	Random Oversampling	GAN	Image Contrast	CLAHE	Filter	Optimizer	Learning Rate	Epochs	Accuracy	Loss	AUC
VGG16_1	✓	✓		✓			Adam	0,001	50	0.81	0.45	0.82
VGG16_2	✓	✓		✓			Adam	0,001	15	0.83	0.39	0.83
VGG16_3	✓	✓		✓			Adam	0,003	15	0.86	0.37	0.85
VGG16_4	✓	✓		✓			RMSprop	0,001	15	0.76	0.46	0.74
VGG16_5	✓	✓		✓			Adam	0,001	15	0.80	0.52	0.79

The model VGG16_5 has been obtained cutting the VGG16 network one level before the last.

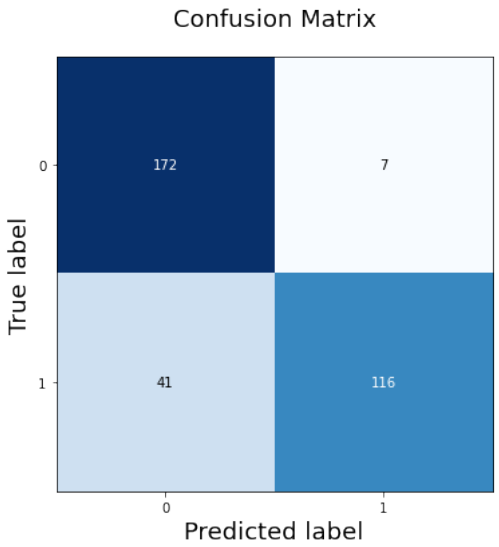
The plot of performance of the better model, VVG16 _3, is the following:

Figure 21: VGG16_V5 Performance



And this is its confusion matrix:

Figure 22: VGG16_V5 Confusion Matrix



5.3.2 InceptionV3

The comprehensive architecture used is the same used in the previous task, with just the exchange of the pretrained CNN used:

Figure 23: Mass vs Calcification InceptionV3 Architecture

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
random_flip (RandomFlip)     (None, 150, 150, 1)       0
_________________________________________________________________
random_rotation (RandomRotat (None, 150, 150, 1)       0
_________________________________________________________________
conv2d_94 (Conv2D)           (None, 150, 150, 3)       6
_________________________________________________________________
inception_v3 (Functional)    (None, 3, 3, 2048)        21802784
_________________________________________________________________
flatten (Flatten)            (None, 18432)             0
_________________________________________________________________
dense (Dense)                (None, 512)               9437696
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 31,240,999
Trainable params: 9,438,215
Non-trainable params: 21,802,784
_________________________________________________________________
```

The hyperparameters on which we performed tries were the same of the previous task:

- the level to cut
- the optimizer
- the learning rate
- the batch size

Here are a results of the better results:

	Data Augumentation	Random Oversampling	GAN	Image Contrast	CLAHE	Filter	Optimizer	Learning Rate	Epochs	Accuracy	Loss	AUC
InceptionV3 _1	✓	✓		✓			Adam	0,001	50	0.88	0.34	0.88
InceptionV3 _2	✓	✓		✓			Adam	0,001	20	0.89	0.25	0.90
InceptionV3 _3	✓	✓		✓			Adam	0,001	15	0.88	0.30	0.88
InceptionV3 _4	✓	✓		✓			Adam	0,003	20	0.88	0.32	0.88
InceptionV3 _5	✓	✓		✓			RMSProp	0,001	20	0.87	0.37	0.87
InceptionV3 _6	✓	✓		✓			RMSProp	0,001	20	0.87	0.28	0.87

The model InceptionV3 _6 has been obtained cutting the InceptionV3 network one level before the last.

The plot of performance of the better model is the following:

And this is its confusion matrix:

Figure 24: InceptionV3_2 Performance

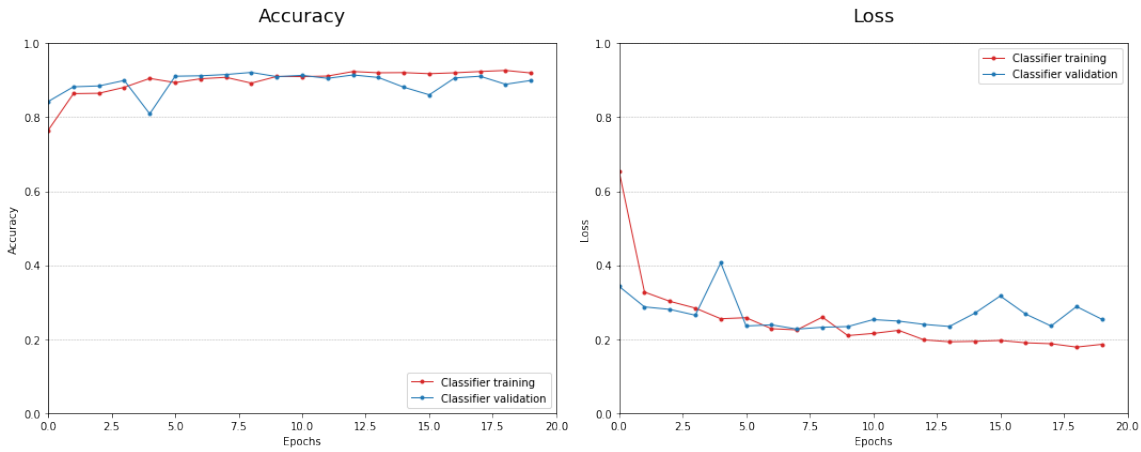
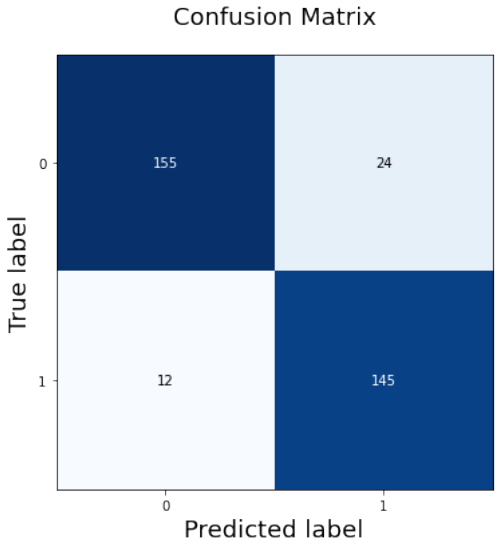


Figure 25: InceptionV3 _2 Confusion Matrix



5.4 Benign vs Malignant

5.4.1 VGG16

The architecture used for the Benign vs Malignant task is the same used for Mass vs Calcification task.

The hyperparameters on which we performed tries where:

- the level to cut
- the optimizer
- the learning rate
- the batch size

The data augmentation step was performed in all the best tries so we don't list the column in the table.

Here are a results of the better results obtained by varying the hyperparameters:

Figure 26: Benign vs Malignant VGG16 Architecture

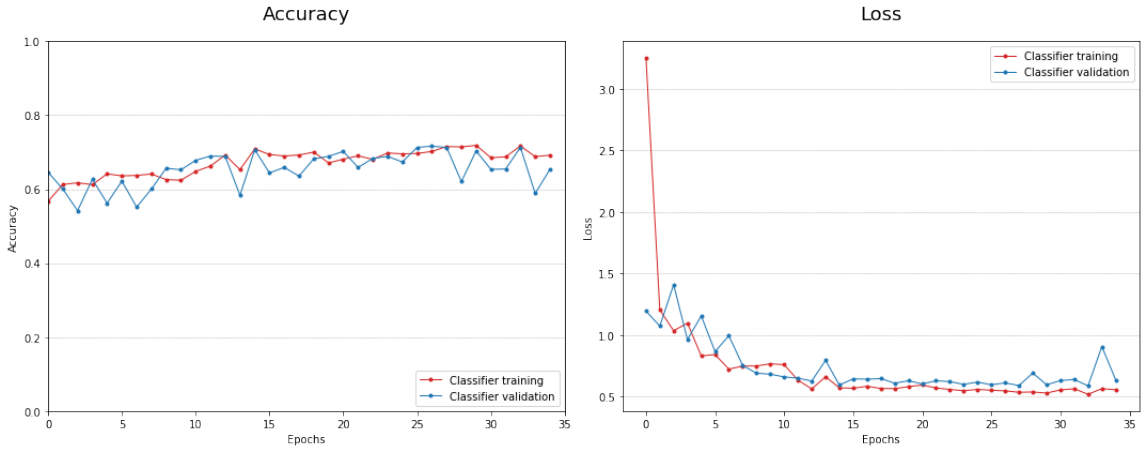
Layer (type)	Output Shape	Param #
random_flip (RandomFlip)	(None, 150, 150, 1)	0
random_rotation (RandomRotat	(None, 150, 150, 1)	0
conv2d (Conv2D)	(None, 150, 150, 3)	6
vgg16 (Functional)	(None, 4, 4, 512)	14714688
flatten (Flatten)	(None, 8192)	0
dense (Dense)	(None, 512)	4194816
dense_1 (Dense)	(None, 1)	513

Total params: 18,910,023
Trainable params: 4,195,335
Non-trainable params: 14,714,688

	Rand. Over-samp.	Cut	Image Con-trast	Optimize	Learning Rate	Epochs	Accuracy	Loss	AUC	F2
VGG16_1	✓	block5	✓	Adam	0,001	30	0.58	0.67	0.62	0.51
VGG16_2	✓	block4	✓	Adam	0,001	35	0.70	0.60	0.62	0.83
VGG16_3	✓	block4	✓	RMSprop	0,001	30	0.58	0.63	0.60	0.57

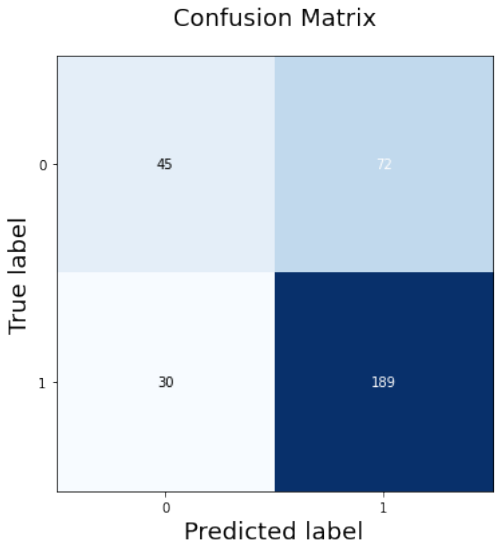
The plot of performance of the better model is the following:

Figure 27: VGG16 _V2 Performance



The result of the optimal model can be seen in its confusion matrix

Figure 28: VGG16 _2 Confusion Matrix



5.4.2 InceptionV3

The comprehensive architecture used is the same used in the previous task, with just the exchange of the pretrained CNN and the addition of a Droput:

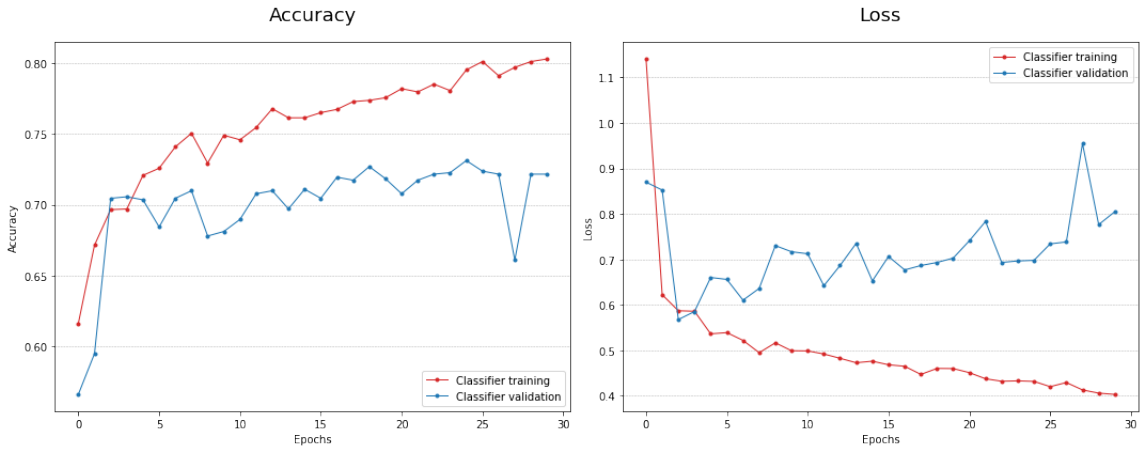
Figure 29: Benign vs Malignant InceptionV3 Architecture

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
random_flip (RandomFlip)     (None, 150, 150, 1)       0
_________________________________________________________________
random_rotation (RandomRotat (None, 150, 150, 1)       0
_________________________________________________________________
conv2d_94 (Conv2D)           (None, 150, 150, 3)       6
_________________________________________________________________
inception_v3 (Functional)    (None, 3, 3, 2048)        21802784
_________________________________________________________________
flatten (Flatten)            (None, 18432)             0
_________________________________________________________________
dropout (Dropout)            (None, 18432)             0
_________________________________________________________________
dense (Dense)                (None, 512)               9437696
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 31,240,999
Trainable params: 9,438,215
Non-trainable params: 21,802,784
_________________________________________________________________
```

The Dropout layer, added in this case, is necessary in order to avoid the overfitting

problem we faced in the task. Here are an example of the training performance of the network without Dropout:

Figure 30: Benign vs Malignant InceptionV3 Overfitting



The hyperparameters on which we performed tries were the same of the previous task:

- the level to cut
- the optimizer
- the learning rate
- the batch size

Here are a results of the better results:

	Random Over-sampling	Cut	Image Contrast	Optimiz	Learning Rate	Epochs	Accurac	Loss	AUC	f2
InceptionV3_1	✓	Top	✓	Adam	0,001	15	0.68	0.59	0.69	0.67
InceptionV3_2	✓	Top	✓	RMSProp	0,001	15	0.70	0.58	0.68	0.72
InceptionV3_3	✓	Top+1	✓	RMSProp	0,001	15	0.68	0.59	0.68	0.70

The plot of performance of the better model is the following:
And this is its confusion matrix:

Figure 31: InceptionV3_2 Performance

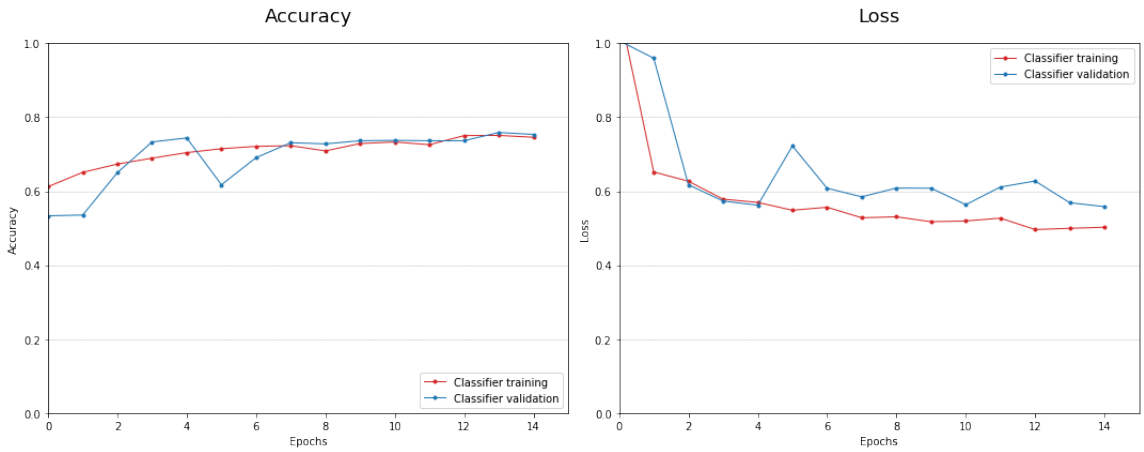
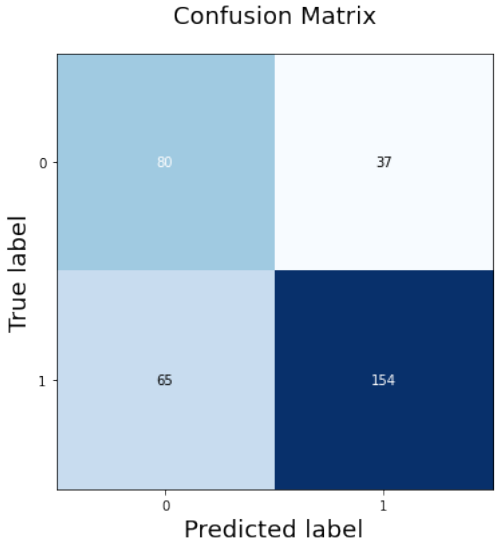


Figure 32: InceptionV3_2 Confusion Matrix



6 Siamese Network

A Siamese neural network is a type of neural network that uses the same architecture to process two different input, producing as output two features that usually are compared in order to evaluate their similarity.

A complete treatment of this kind of network in the computer vision task can be found at [5].

As well documented, a standard approach to deal with this network is to evaluate the difference from two set of features both extracted from two twin networks, in order to classify, for example, the images as belonging to the same class or not.

In our case we use this network in order to process the abnormalities and the baselines, then instead of comparing the two output we concatenate them. We do this concatenation in order to improve and calibrate our networks for the classification task and not to asses if the two images belong to the same label (that is our hypothesis).

In fact if we evaluate the difference from the two set of features we risk to lose a lot of important information about the abnormalities that the MLP classifier need to take in

consideration to correctly answer to the problem.

Instead, using a concatenation we don't lose any information while also providing the classifier with an additional set of information useful to calibrate and balance the system.

As core architecture we start with the one used in the scratch task.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 148, 148, 32)	320
max_pooling2d (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_1 (Conv2D)	(None, 72, 72, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 36, 36, 64)	0
conv2d_2 (Conv2D)	(None, 34, 34, 128)	73856
conv2d_3 (Conv2D)	(None, 32, 32, 128)	147584
max_pooling2d_2 (MaxPooling2	(None, 16, 16, 128)	0
flatten (Flatten)	(None, 32768)	0
dense (Dense)	(None, 512)	16777728
dense_1 (Dense)	(None, 1)	513

Total params: 17,018,497
Trainable params: 17,018,497
Non-trainable params: 0

In order to provide the system with the corresponding and correct data, at the first step we perform a dataset transformation task. In this task each couple Baseline image, Abnormality image is paired and associated with the corresponding abnormality's label.

So if the couple Baseline, Abnormality was referred to element of class 0 (Baseline) , 1 (Mass, Benign), the couple is paired and associated to the class 1 (Mass, Benign). Finally the classes are translated again in the standard (0, 1) form, as for the other steps, namely (Mass, Calcification).

The network receives in input a couple of images Baseline, Abnormality and set as output 0 if the Abnormality was a Mass, 1 if it was a Calcification.

The global network architecture can be seen as:

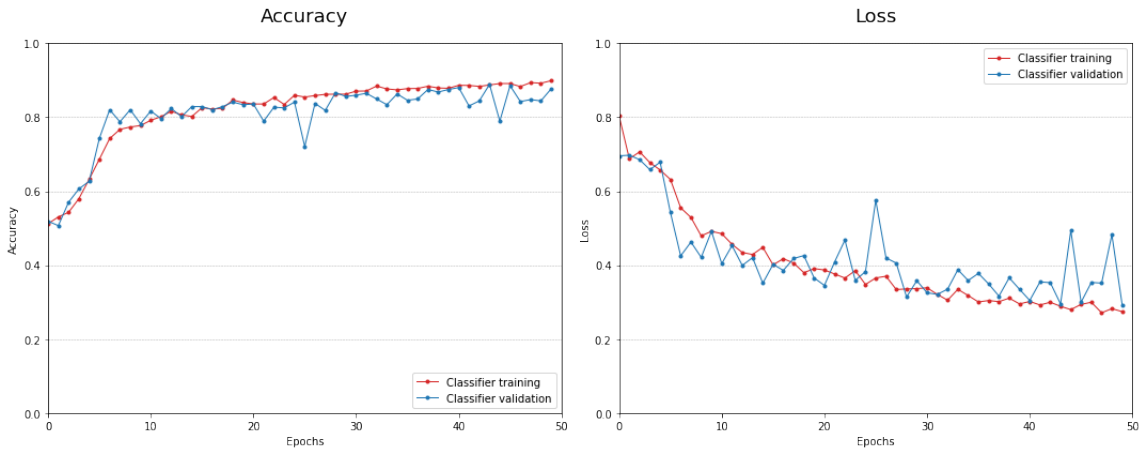
The performances of the models trained and tested can be seen above:

	Data Agu-ment.	Random Overs.	Image Con-trast	Optimiz	Learnin Rate	Epochs	Accurac	Loss	AUC
Siamese_1	✓	✓	✓	Adam	0,001	50	0.84	0.37	0.84
Siamese_2	✓	✓	✓	Adam	0,001	30	0.79	0.49	0.79
Siamese_3	✓	✓	✓	RMSprop	0,001	50	0.86	0.32	0.85
Siamese_4	✓	✓	✓	RMSprop	0,001	30	0.83	0.43	0.83

```
Model: "Siamese-Network"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
Abnormalities (InputLayer)      [(None, 150, 150, 1) 0
__________________________________________________________________________________________________
Baselines (InputLayer)          [(None, 150, 150, 1) 0
__________________________________________________________________________________________________
Siamese (Functional)            (None, 512)          17017984    Abnormalities[0][0]
                                                                 Baselines[0][0]
__________________________________________________________________________________________________
Concatenation (Concatenate)     (None, 1024)         0           Siamese[0][0]
                                                                 Siamese[1][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 1024)         0           Concatenation[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 512)          524800      dropout[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 1)            513         dense_1[0][0]
==================================================================================================
Total params: 17,543,297
Trainable params: 17,543,297
Non-trainable params: 0
__________________________________________________________________________________________________
```

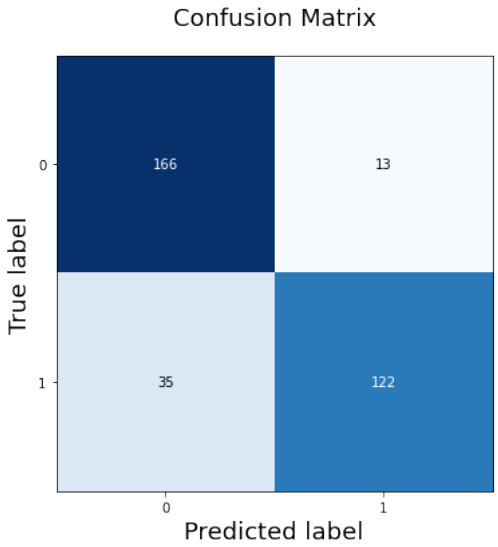
The plot of performance of the better model is the following:

Figure 33: Siamese_3 Performance



And this is its confusion matrix:

Figure 34: Siamese 3 _Confusion Matrix



7 Ensemble

For the Ensemble task we decided to use the Benign vs Malign models obtained during the previous tasks in order to obtain better performance overall.

We decided to use:

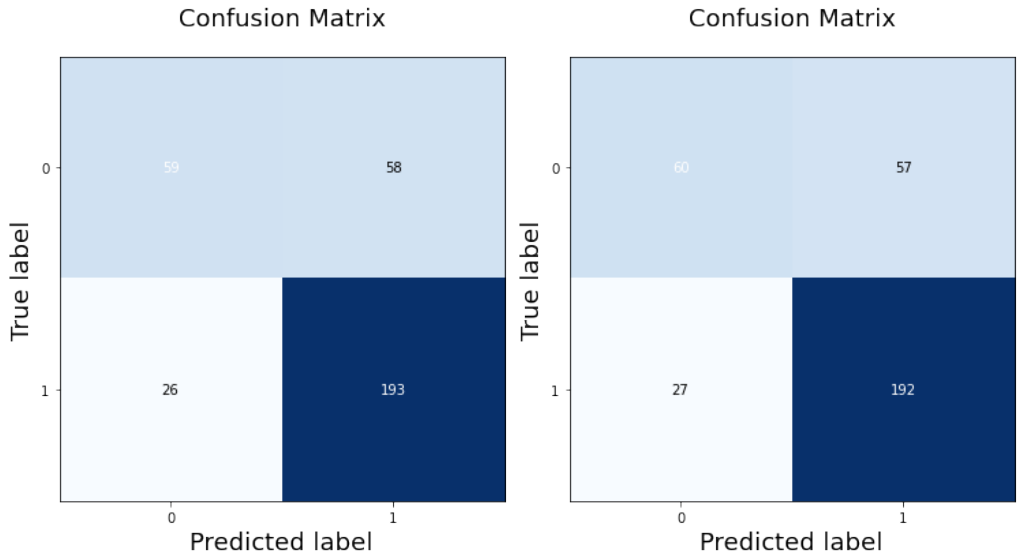
- The model obtained with the Scratch task
- The model with the pretrained InceptionV3 network
- The model with the pretrained VGG16 network

We predicted the test set labels by combining the test set labels prediction of the three models by aggregating the results using an average voting and majority voting. We then computed the metrics comparing the labels obtained by each strategy of voting and the true labels. From the tables we can clearly see that the ensembling of the three models,independently form the voting strategy, is able to notably improve the metrics.

	Accuracy	*F2*
Best_BM_Scratch	0.71	0.84
Best_BM_VGG16	0.70	0.83
Best_BM_InceptionV3	0.70	0.72
Ensemble_Average	0.88	0.86
Ensemble_Majority	0.88	0.85

In the confusion matrix we can see how, with respect to the confusion matrices of the *Scratch* and *VGG16*, we have been able to better detect the True Malign, therefore decreasing the tuples incorrectly labeled as Malign. This is not the case with the *InceptionV3*, that however has a significant difference in number of tuples Benign but incorrectly labeled as Malignant.

Figure 35: Confusion Matrices



(a) Average voting

(b) Majority voting

8 Conclusions

From the analysis and results we have obtained, we were able to establish that the data set is well adapt to the mass-calcification problem.

On the other hand, we had a lot of problems in developing a satisfactory model for the benign-malignant problem.

The application of filters able to identify and stress the main form of the abnormalities, and the development of a GAN able to create sythetic abnormalities, are necessary steps that have to be carried out in order to obtain at least decent results.

This approach however can be applied only to CNN trained from scratches, since it's unlikely that pre-trained CNNs have been detected with such treated images.

It is also important to notice that during numerous trainings, the random splitting of the original training set into the effective training set and validation set caused fluctuation of the performance, in particular for the Benign vs Malignant classification, for the worse, probably caused by the unbalance between the two sets.

Numerous tries had to be done for each configuration in order to find out if the first result was just a case or the norm.

References

[1] Richa Agarwal et al. "Automatic mass detection in mammograms using deep convolutional neural networks". In: *Journal of Medical Imaging* 6.3 (2019), p. 031409.

[2] Lenin G Falconı, Marıa Pérez, and Wilbert G Aguilar. "Transfer learning in breast mammogram abnormalities classification with mobilenet and nasnet". In: *2019 International Conference on Systems, Signals and Image Processing (IWSSIP)*. IEEE. 2019, pp. 109–114.

[3] LG Falconı et al. "Transfer learning and fine tuning in breast mammogram abnormalities classification on CBIS-DDSM database". In: *Advances in Science, Technology and Engineering Systems* 5.2 (2020), pp. 154–165.

[4] Shuyue Guan and Murray Loew. "Breast cancer detection using synthetic mammograms from generative adversarial networks in convolutional neural networks". In: *Journal of Medical Imaging* 6.3 (2019), p. 031411.

[5] Abhilash Nandy et al. "A Survey on Applications of Siamese Neural Networks in Computer Vision". In: *2020 International Conference for Emerging Technology (INCET)*. 2020, pp. 1–5. DOI: 10.1109/INCET49848.2020.9153977.

[6] Dina A Ragab et al. "Breast cancer detection using deep convolutional neural networks and support vector machines". In: *PeerJ* 7 (2019), e6201.

[7] Wessam M Salama and Moustafa H Aly. "Deep learning in mammography images segmentation and classification: Automated CNN approach". In: *Alexandria Engineering Journal* 60.5 (2021), pp. 4701–4709.

[8] Li Shen et al. "Deep learning to improve breast cancer detection on screening mammography". In: *Scientific reports* 9.1 (2019), pp. 1–12.

[9] Bartosz Swiderski et al. "Deep neural system for supporting tumor recognition of mammograms using modified GAN". In: *Expert Systems with Applications* 164 (2021), p. 113968.

[10] Jaime Zabalza et al. "Novel Two-Dimensional Singular Spectrum Analysis for Effective Feature Extraction and Data Classification in Hyperspectral Imaging". In: *IEEE Transactions on Geoscience and Remote Sensing* 53.8 (2015), pp. 4418–4433. DOI: 10.1109/TGRS.2015.2398468.