Convolutional neural network for Medical Imaging Analysis
Abnormality detection in mammography

Antonio Acquavia
Matilde Mazzini

February 2021

1 Introduction

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

- Benignant-Malignant classification
- Mass-Calcification classification

The dataset usually used is CBIS DDSM: Curated Breast Imaging Subset of Digital Database for Screening Mammography.
The dataset represents a collection of high resolution images in DICOM format. In the original collection, for each image there are different information, including a binary mask that locates the abnormality.
In this case the abnormality patch has already been extracted from the original image according to the binary mask.
In particular, for our project the dataset is composed by a pair of NumPy files for the training set and a pair for the test set.
First of all we analyzed the dataset provided, viewing the images, analyzing the format and observing the distribution of the labels.
Furthermore, we also looked for the work already carried out on problems like this, to find ideas and understand the possible difficulties that we might encounter.
To solve these problems we tried to build networks from scratch and to use pre-trained networks. Afterwards, we used the network built from scratch to address the mass-calcification problem with a Siamese Architecture. Finally, we tried to obtain better results for the benignant-malignant problem using an ensamble of the best obtained classifiers.
During the preprocessing phases we realized that many transformations applied to the data were repeated, for this reason we have grouped all these operations in a single notebook called 'utility.ipynb'.

2 Task 1: Papers

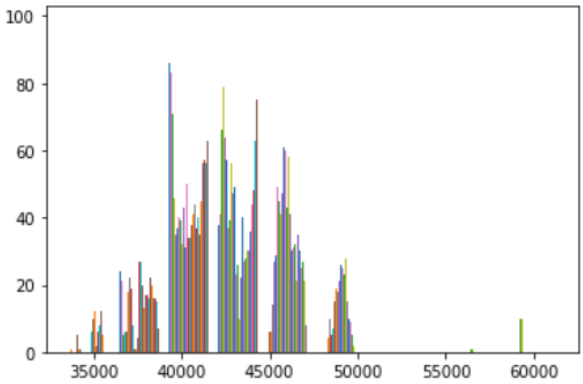
Before starting to work on the problem, we examined the works carried out from this data set. Consulting the papers, we learned several useful techniques to apply on our problem.
The list of the papers is in the references section.
To address the Pretrained network tasks we saw that the most popular networks found in the documentations were:

- VGG16 [6][7]
- InceptionV3 [7]
- ResNet50 [6][7]
- AlexNet [5][7]

For our project we decided to use VGG16 and InceptionV3 networks.
Besides, in paper[5] the authors added a new SVM classifer on top of the last convolutional layer of the pretrained network, in order to exploit the network as a features extractor removing the fully connected layer on top of it. Having a small dataset, we decided to use this technique too, but instead of giving the extracted features to a new SVM classifier we used a MLP classfier. We considered the level at which we cut the network to extract features as an hyper parameter.
The papers [7][5] had to face different preprocessing problems with respect to us, they had for example to extract the region of interest patch from the overall image before giving it as input to the network. Seeing all the preprocessing steps made by the authors, we decided to follow an interesting one: they used the CLAHE algorithm to enhance the contrast of each image in order to obtain better results.
To face overfitting one of the most adopted technique of the papers[5][7] was data augmentation. Data is often augmented by applying horizontal flip, rotation and cropping.

3 Dataset Visualization and Preparation

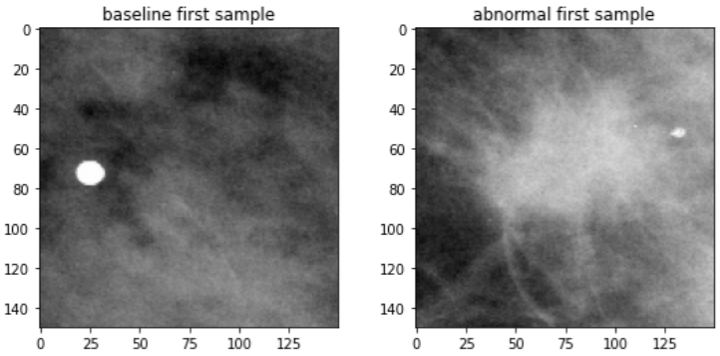
As already said, the data set that was provided for this project is different from the original CBIS-DDSM, because some pre-processing to extract the abnormality patches was already performed on it. In the data set that we used, images and labels are given as NumPy arrays. The NumPy images provided are 16-bit gray-scale images, this means that they are single channel images with pixel values going from 0 to 65536.
Histogram of the pixel values of a sample image:



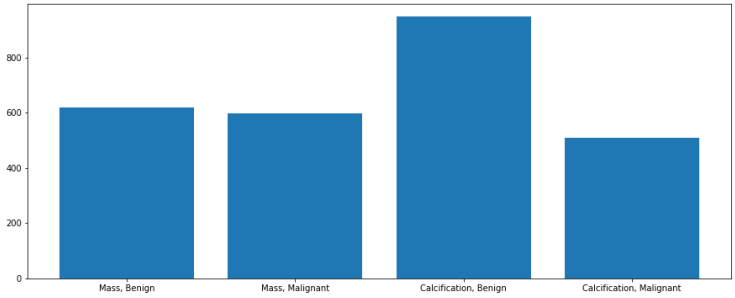
For each patient the dataset offers both a Baseline (healty tissue) patch and an Abnormality patch. The Baseline patch are located in the even positions of the images array and the corresponding Abnormality patch is in the consequent odd index of the same images array.
In the same index of each image we can find the corresponding label in the labels array.
The labels are:

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

Example of Baseline patch (label 0) and corresponding Abnormal Mass-Malignant (label 2) patch:



Training Labels Distribution:



For each task we had to build a different data set starting from the original one modifying also the labels depending on the specific task. We had to build three different data sets:

- Benignant vs Malignant data set
- Mass vs Classification data set
- Siamese Network data set

3.1 Benignant vs Malignant data set creation

In the Benignant vs Malignant classification problem we selected only the Abnormal patches removing both the Baseline images from the images array and the labels in even positions in the labels array. We then selected the labels of each Benignant patch, those equal to 1 and 3 and set those labels to 0. We did the same for the Malignant patches, setting to 1 the labels equal to 2 and 4.

3.2 Mass vs Calcification data set creation

In the Mass vs Calcification classification problem we selected only the Abnormal patches removing both the Baseline images from the images array and the labels in even positions in the labels array. We then selected the labels of each Mass patch, those equal to 1 and 2 and set that label to 0. We did the same for the Calcification patches, setting to 1 the labels equal to 3 and 4.

3.3 Siamese data set creation

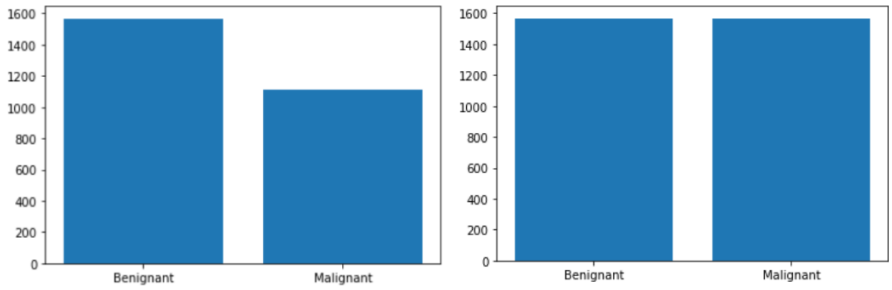
In the Siamese Network we had to build a data set made by an array of images tuples and an array of labels, one for each tuple. Each tuple is composed by a a Baseline patch and its corresponding Abnormal patch. The label of the tuple has to address the Abnormality type: if the Abnormal patch is a Mass, the label of the tuple is set to 0, if the Abnormal path of the tuple is a Calcification, the label is set to 1.
Example of a data set entry:

$$(\text{Baseline, Mass}) \longrightarrow 0$$

$$(\text{Baseline, Calcification}) \longrightarrow 1$$

3.4 Class Balancing

We can see that in both Mass VS Calcification and Benignant VS Malignant problem the classes are slightly imbalanced, we performed both undersampling and oversampling using random sampling on the data sets and performed training tests with both balanced data sets.
Oversampling example on Benignant-Malignant data set:

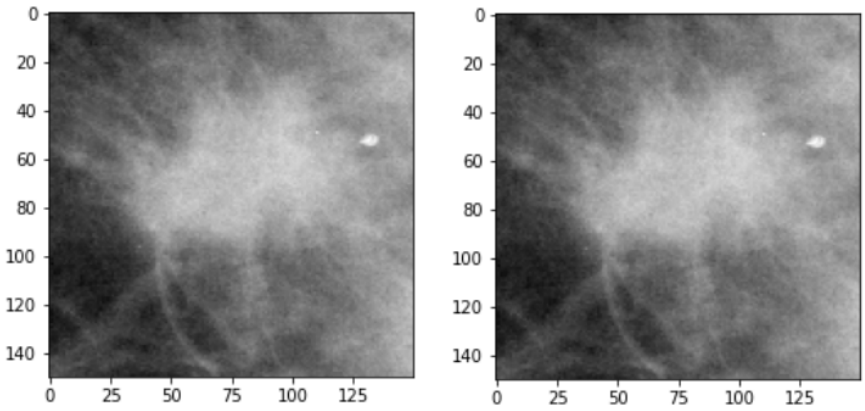


3.5 Preprocessing

3.5.1 Contrast

Reading the papers we found out that to improve image quality they enhanced image contrast by using the CLAHE algorithm. We performed some test applying the tensorflow adjust_contrast function to the training images to see if this

lead to better results.
Example of before and after the application of contrast with a contrast_factor=2:



3.5.2 Normalization

Before giving the images in input to the CNNs, we applied normalization on the pixel values transforming them from values: [0,65536] to values:[0,1].

3.5.3 Dataset Splitting

Having available the training set and the test set, to proceed with the next steps we had to obtain the validation set. To do this we used a function of the sklearn library on the training set, setting the splitting factor to 0.2. This function executes also the shuffle before the split.

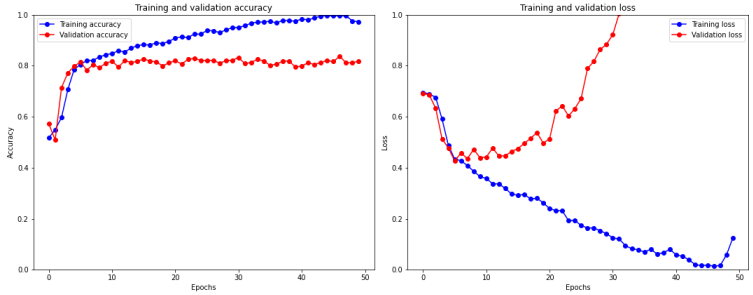
4 CNN from Scratch

4.1 Mass-Calcification

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

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 148, 148, 32)      320
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 15, 15, 128)       147584
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 7, 7, 128)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 6272)              0
_________________________________________________________________
dense_4 (Dense)              (None, 512)               3211776
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 513
=================================================================
Total params: 3,452,545
Trainable params: 3,452,545
Non-trainable params: 0
_________________________________________________________________
```

In the pre-processing phase, the images has been normalized in a range from 0 to 1.
Since the classes were unbalanced, we tried oversampling and undersampling techniques based on random sampling: we obtained better results with oversampling.
The first result was not so good, we faced a severe overfitting:



This can be seen both from the chart and from the measurements obtained on the validation set:

	Training Set	Validation Set
Accuracy	0,97	0,81
Loss	0,10	1,55

To overcome this problem, we used Data Augmentation because in the literature it produced good results, in particular we applied:

- horizontal flipping
- 90° rotation

Besides the Data Augmentation, we tried to improve our results tuning the hyper-parameters, especially we focused on:

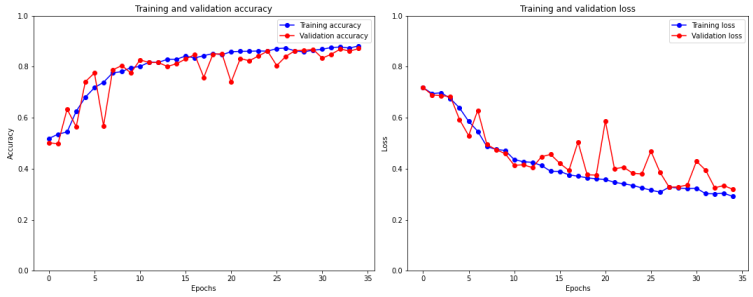
- learning rate
- optimizers
- number of epochs

Here we have a summary of the best tests:

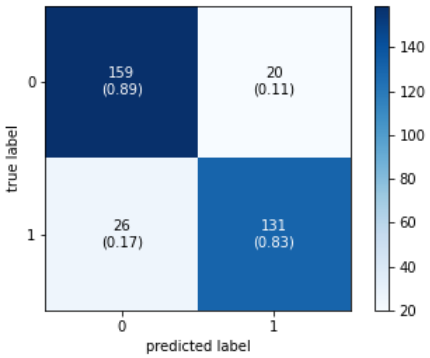
Scratch

	Data Augmentation	Optimizer	Learning Rate	Epochs	Accuracy	Loss	AUC
SCRATCH_1	☐	Adam	0,001	50	-	-	-
SCRATCH_2	☑	Adam	0,001	50	0,82	0,39	0,81
SCRATCH_3	☑	Adam	0,001	50	0,85	0,32	0,85
SCRATCH_4	☑	RMSprop	0,001	50	0,87	0,29	0,87
SCRATCH_5	☑	RMSprop	0,003	50	0,85	0,47	0,85
SCRATCH_6	☑	RMSprop	0,001	35	0,87	0,31	0,87

The best model obtained is SCRATCH_6, although in the central phase the fluctuations are quite evident, the validation loss curve stabilizes with the passing of the epochs. This was verified in several tests, in which, with a greater number of epochs, the results stabilized around 0.87 accuracy and the validation loss curve remained close to the training loss curve. For this reason we felt that 35 was a satisfactory number for the epochs instead of 50.



The confusion matrix fully reflects the goodness of the results:



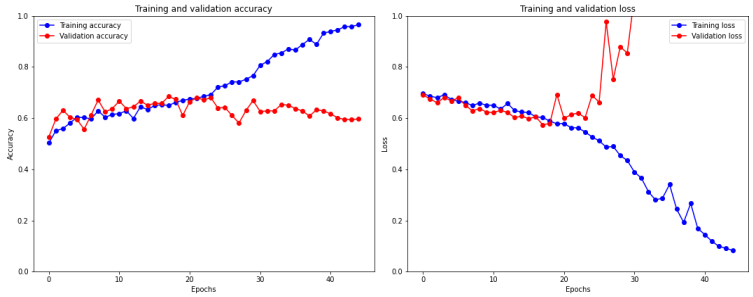
4.2 Benignant-Malignant

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

Layer (type)	Output Shape	Param #
conv2d_8 (Conv2D)	(None, 148, 148, 32)	320
max_pooling2d_8 (MaxPooling2	(None, 74, 74, 32)	0
conv2d_9 (Conv2D)	(None, 72, 72, 64)	18496
max_pooling2d_9 (MaxPooling2	(None, 36, 36, 64)	0
conv2d_10 (Conv2D)	(None, 34, 34, 128)	73856
max_pooling2d_10 (MaxPooling	(None, 17, 17, 128)	0
conv2d_11 (Conv2D)	(None, 15, 15, 128)	147584
max_pooling2d_11 (MaxPooling	(None, 7, 7, 128)	0
flatten_2 (Flatten)	(None, 6272)	0
dense_4 (Dense)	(None, 512)	3211776
dense_5 (Dense)	(None, 1)	513

Total params: 3,452,545
Trainable params: 3,452,545
Non-trainable params: 0

In the pre-processing phase, the images has been normalized in a range from 0 to 1.
The first test was executed without appyling any constrast, 45 epochs, and adam optimizer. As we can see from this plot we faced overfitting:



This can be seen both from the chart and from the measurements obtained on the validation set:

	TRAINING SET	VALIDATION SET
ACCURACY	0,95	0,59
LOSS	0,08	3,41

To overcome overfitting, we first reduced the number of epochs, and inspired by literature we used Data Augmentation, in particular we applied only the 90° degreee rotation because it is written that other angles can cause distortion to the images. Besides the Data Augmentation, we then tried to improve our results applying the enhancement contrast technique and tuning the hyper-parameters, especially we focused on:

- optimizers
- number of epochs
- different data augmentation combinations between rotation and horizontal flip.

Here we have a summary of the most relevant tests:

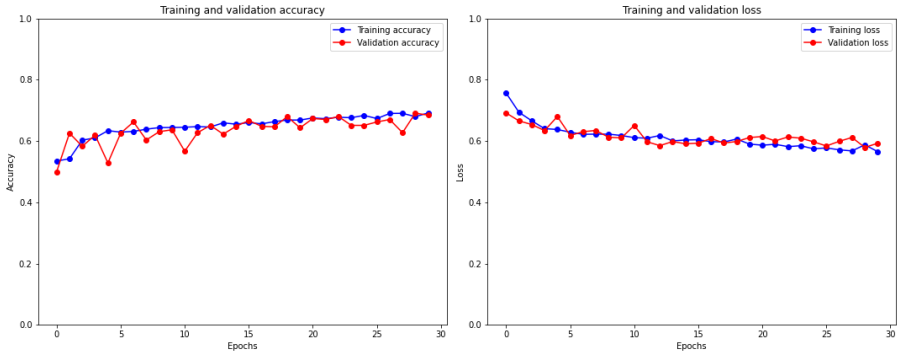
Scratch

	Data Augmentation	Optimizer	Learning Rate	Contrast	Epochs	Accuracy	Loss	F2	AUC
SCRATCH_BM_V1	☐	ADAM	0,001	☐	45	-	-	-	-
SCRATCH_BM_V2	✓	ADAM	0,001	2	30	0,57	0,70	0,68	0,62
SCRATCH_BM_V3	✓	RMSPROP	0,001	2	30	0,61	0,63	0,65	0,64
SCRATCH_BM_V4	✓	RMSPROP	0,001	☐	30	0,6	0,65	0,59	0,61
SCRATCH_BM_V5	✓	ADAM	0,001	☐	30	0,58	0,68	0,63	0,61
SCRATCH_BM_V6	✓	RMSPROP	0,001	2	30	0,65	0,61	0,61	0,65

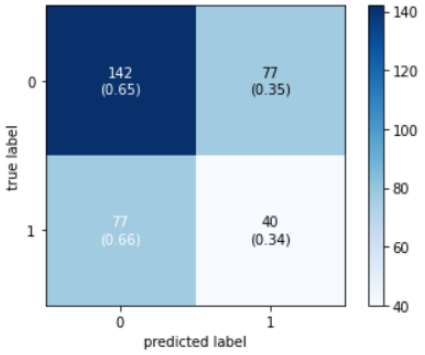
To evaluate our models, we decided to not consider only the accuracy, but also the f2 score metric [3]. In medical imaging classification problems is important to keep an eye on this specific metric: considering that we are working with breast cancer, a prediction that states the absence of cancer while the ground truth states the opposite leads to huge consequences in real life. Having Malignant samples labeled as 1 (positive samples), the described case is represented in the confusion matrix by the False Negatives:

- Negative prediction: Benignant
- Positive truth: Malignant

To give more importance to False Negatives we should consider recall more important than precision, to this, F2 is a metric that considers recall to be twice as important as precision.
The best model obtained is SCRATCH_BM_6:



We have choosen as best model the one that gives the best trade-off between accuracy and f2, despite this, as we can see from the confusion matrix, the results obtained are not very satisfactory:



5 Pre-trained CNN

After building new networks from scratch, we started to train pre-trained neural networks. Relying on literature, we decided to try the following architectures:

- VGG16
- InceptionV3

Always following the literature, we decided to use existing networks to obtain features on which to train a classifier. We used the pretrained networks as feature extractors after removing the fully connected layers on top of it. We then applyed a new MLP classifier on top.
Taking advantage of the experience gained in scratch networks, we started already by carrying out data augmentation and applying the contrast (1.5 or 2).

5.1 Pre-Processing

In the pre-processing phase, the images has been normalized in a range from 0 to 1.
Since the classes were unbalanced, we tried oversampling and undersampling techniques based on random sampling: we obtained better results with oversampling, especially in the Benignant-Malignant case.

5.2 From 1-channel to 3-channel images

The pre-trained architectures requires to work with 3-channel images, but our dataset is composed of gray-scale images.
In the papers, the image is replicated on 3 channels before giving it as imput to the pretrained model. Instead, we experimented the use of a convolutional layer before the pre-trained network to enlarge the depth from 1 to 3. To do this, our convolutional layer must have 3 filters of size 1x1.

Idea The idea is to make the transition from 1-channel to 3-channel image trainable, instead of applying simple repetition.

We did the comparison between the two solutions on the best configuration obtained for the benignant vs malignant problem.

	Accuracy	Loss	F2	AUC
Replication	0,68	0.57	0.72	0.71
Convolutional Layer	0,69	0,58	0,69	0,70

We can observe that there is a not too marked difference between the two approaches.

Of course, proving that one is better than the other would require a statistically significant number of tests. In this case we have limited ourselves only to comparing the two best results for a given configuration, in order to show that in our case they are quite interchangeable.

5.3 Mass-Calcification

5.3.1 VGG16

The architecture used in this task is the following, we can see that before the VGG16 layer, a convolutional layer that increments the depth of the input layer is added. We can also see the data augmentation layers.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
random_flip_7 (RandomFlip)   (None, 150, 150, 1)       0
_________________________________________________________________
random_rotation_7 (RandomRot (None, 150, 150, 1)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 150, 150, 3)       6
_________________________________________________________________
vgg16 (Functional)           (None, 4, 4, 512)         14714688
_________________________________________________________________
flatten_7 (Flatten)          (None, 8192)              0
_________________________________________________________________
dense_14 (Dense)             (None, 512)               4194816
_________________________________________________________________
dense_15 (Dense)             (None, 1)                 513
=================================================================
Total params: 18,910,023
Trainable params: 4,195,335
Non-trainable params: 14,714,688
_________________________________________________________________
```

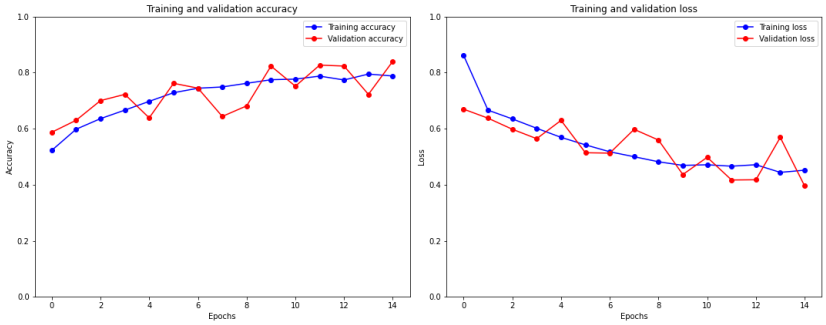
In this training, the hyper-parameters on which we have focused were:

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

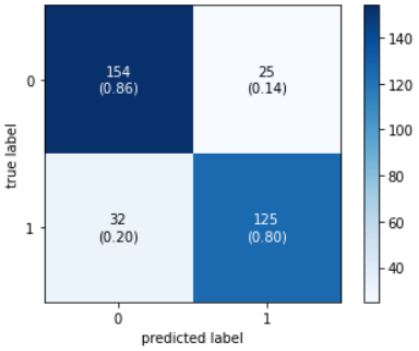
We tryed to cut the pretrained network at different levels to see if the representation extracted could be better classified when extracted from previous layers. We started the tests cutting only the latest layers, going on with tests with higher cut levels we saw that this was not leading to better results.
In some tests the data augmentation is composed of just the rotation layer, without the horizontal flip. Here are listed the best results. The best result was obtained using oversampling.

	Data Augmentation	Cut	Optimizer	Learning Rate	Epochs	Accuracy	Loss	AUC
VGG16_V1	☐	top	Adam	0,001	60	-	-	-
VGG16_V2	✓	top	Adam	0,001	15	0,81	0,42	0,81
VGG16_V3	✓	top	Adam	0,001	15	0,66	0,65	0,68
VGG16_V6	✓	top+1	Adam	0,001	15	0,82	0,38	0,82
VGG16_V5	✓	top	Adam	0,001	15	0,83	0,41	0,82

In some tests we also performed l2 regularization to solve the bouncing behaviour of the validation we see in the plot of the selected model, without big improvements.
Plot of the 'VGG16_V5' training:



Confusion matrix:



5.3.2 InceptionV3

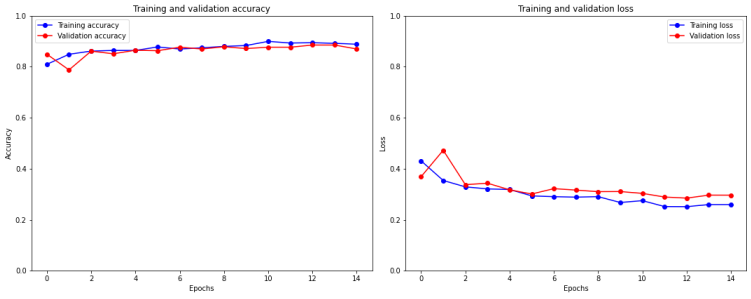
the architecture used in the InceptionV3 Pretrained task is the following, we started with data augmentation to contrast the overfitting that we saw in every task.

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
random_flip_2 (RandomFlip)   (None, 150, 150, 1)       0
_________________________________________________________________
random_rotation_2 (RandomRot (None, 150, 150, 1)       0
_________________________________________________________________
conv2d_96 (Conv2D)           (None, 150, 150, 3)       6
_________________________________________________________________
inception_v3 (Functional)    (None, 3, 3, 2048)        21802784
_________________________________________________________________
flatten_2 (Flatten)          (None, 18432)             0
_________________________________________________________________
dense_4 (Dense)              (None, 512)               9437696
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 513
=================================================================
Total params: 31,240,999
Trainable params: 9,438,215
Non-trainable params: 21,802,784
_________________________________________________________________
```

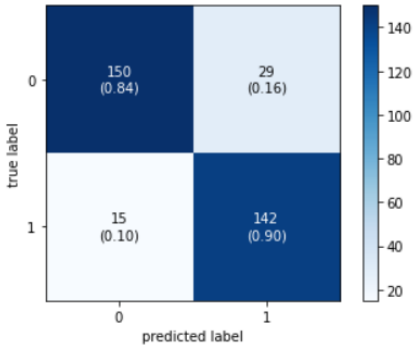
The hyperparameters we focues on were the same of the VGG16 task. In some tests the data augmentation is composed of just the rotation layer, without the horizontal flip. Undersampling was giving overfittig so even in this caase we choosed to use the oversampled data set. Best tests results:

	Data Augmentation	Dropout	Cut	Optimizer	Learning Rate	Epochs	Accuracy	Loss	AUC
INCEPTION_1			top	Adam	0,001	20	-	-	-
INCEPTION_2	✓		top	Adam	0,001	20	0,85	0,46	0,85
INCEPTION_3	✓	✓	top	Adam	0,001	15	0,86	0,36	0,87
INCEPTION_4	✓		top	Rsmprop	0,001	20	0,84	0,35	0,83
INCEPTION_5	✓		top	Adam	0,001	20	0,86	0,32	0,87
INCEPTION_6	✓		top+1	Adam	0,001	20	0,85	0,35	0,85

Training plots of the INCEPTION_3 configuration:



Confusion matrix:



5.4 Beningant-Malignant

5.4.1 VGG16

In this training, the hyper-parameters on which we have focused were:

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

We started from removing just the top from the base network: we observed that there was an improvement in the F2 score but a huge step back from the point of view of accuracy.
Here there is a summary of some of the tests done and the results obtained varying the hyper-parameters above mentioned:

VGG16

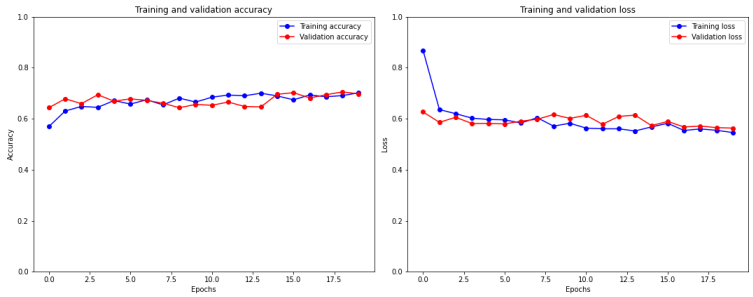
	Data Augmentation	Cut	Contrast	Batch Size	Optimizer	Learning Rate	Accuracy	Loss	F2	AUC
VGG_1	✓	Top	1,5	50	Adam	0,001	0,67	0,68	0,10	0,53
VGG_2	✓	Top	1,5	32	Adam	0,001	0,658	0,61	0,58	0,63
VGG_3	✓	Top + block5	1,5	32	Adam	0,001	0,58	0,66	0,69	0,63
VGG_4	✓	Top	1,5	32	RMSprop	0,001	0,65	0,61	0,44	0,60
BEST VGG	✓	Top	1,5	32	Adam	0,003	0,648	0,63	0,68	0,67

The best architecture used is the following:

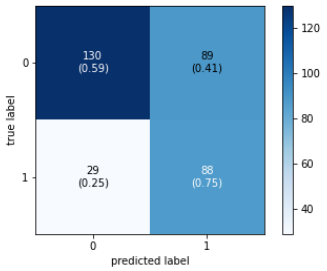
Layer (type)	Output Shape	Param #
random_flip_2 (RandomFlip)	(None, 150, 150, 1)	0
random_rotation_2 (RandomRot	(None, 150, 150, 1)	0
conv2d_2 (Conv2D)	(None, 150, 150, 3)	6
vgg16 (Functional)	(None, 4, 4, 512)	14714688
flatten_2 (Flatten)	(None, 8192)	0
dense_4 (Dense)	(None, 512)	4194816
dense_5 (Dense)	(None, 1)	513

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

These are the accuracy and validation metrics obtained from the training of the model:



Since the problem is a Benignant-Malignant problem, we focused on the number of False Negatives (considering the class Malignant as p): we chose the model that minimized that value.
This result can be observed on the confusion matrix: the number of element in the lower-left cell is very small compared with the others.



5.4.2 InceptionV3

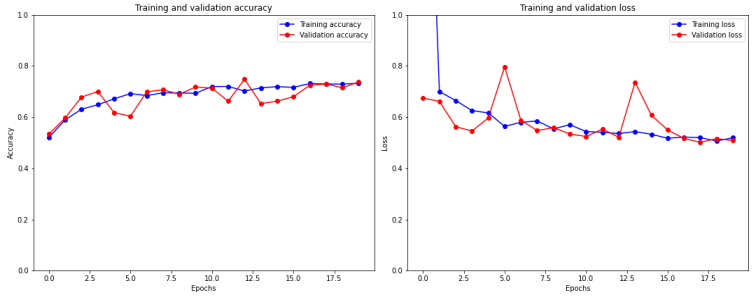
As opposed to VGG-16, in Inception we observed that tuning the cut level gave us some improvements: we trained our best model removing the top level and also the one immediately above.

```
Layer (type)                 Output Shape              Param #
=================================================================
random_flip_1 (RandomFlip)   (None, 150, 150, 1)       0
_________________________________________________________________
random_rotation_1 (RandomRot (None, 150, 150, 1)       0
_________________________________________________________________
conv2d_189 (Conv2D)          (None, 150, 150, 3)       6
_________________________________________________________________
model_1 (Functional)         (None, 3, 3, 1280)        10674848
_________________________________________________________________
flatten_1 (Flatten)          (None, 11520)             0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               5898752
_________________________________________________________________
dense_4 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 257
=================================================================
Total params: 16,705,191
Trainable params: 6,030,343
Non-trainable params: 10,674,848
_________________________________________________________________
```

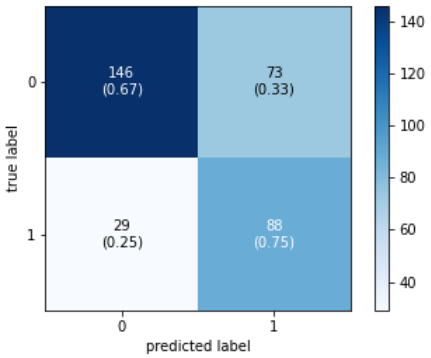
In this case we obtained better results with a further dense layer.
We trained the model for very few epochs because the growth of inception in learning was rapid.
As in all benignant-malignant problems, we have given a great deal of weight to the value of F2,however, taking into consideration the other values.

Inception V3

	Data Augmentation	Dropout	Cut	Optimization	Epochs	Contrast	Accuracy	Loss	F2	AUC
INCEPTION_1	✓		Top	Adam	20	1,5	0,67	0,62	0,55	0,64
INCEPTION_2	✓		Top + 1	Adam	10	1,5	0,70	0,56	0,44	0,64
INCEPTION_3	✓		Top + 1	Adam	10	1,5	0,64	0,62	0,72	0,68
INCEPTION_4	✓		Top + 1	RMSprop	10	1,5	0,70	0,55	0,49	0,65
INCEPTION_5	✓		Top + 1	RMSprop	10	1,5	0,66	0,59	0,69	0,68
INCEPTION_6	✓	✓	Top + 1	RMSprop	20	1,5	0,71	0,56	0,60	0,68
INCEPTION_7	✓		Top + 2	Adam	20	1,5	0,68	0,60	0,19	0,56
BEST INCEPTION	✓		Top + 1	Adam	20	1,5	0,69	0,58	0,69	0,70



The confusion matrix shows us how choosing a good F2 model results in a low number of malignant cases classified as benignant:


146
(0.67)
73
(0.33)
29
(0.25)
88
(0.75)
0
1
true label
0
1
predicted label
140
120
100
80
60
40

6 Task 4: Siamese Architecture

We tested a Siamese Architecture to make a sort of model calibration based on normal tissue patches, for this, instead of calculating the distances between the outputs of the Siamese we concatenated them. To build the Siamese Network architecture we started from the one used in the Scratch task.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_10 (InputLayer)        [(None, 150, 150, 1)]     0
_________________________________________________________________
random_rotation_9 (RandomRot (None, 150, 150, 1)       0
_________________________________________________________________
conv2d_36 (Conv2D)           (None, 148, 148, 32)      320
_________________________________________________________________
max_pooling2d_36 (MaxPooling (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_37 (Conv2D)           (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_37 (MaxPooling (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_38 (Conv2D)           (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_38 (MaxPooling (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_39 (Conv2D)           (None, 15, 15, 128)       147584
_________________________________________________________________
max_pooling2d_39 (MaxPooling (None, 7, 7, 128)         0
_________________________________________________________________
flatten_9 (Flatten)          (None, 6272)              0
_________________________________________________________________
dense_29 (Dense)             (None, 512)               3211776
=================================================================
Total params: 3,452,032
Trainable params: 3,452,032
Non-trainable params: 0
_________________________________________________________________
```

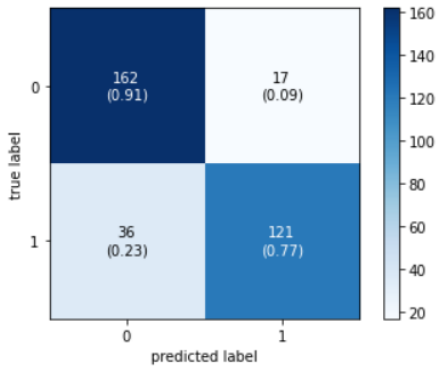
As said in the data set preprocessing paragraph, during the training we give as input to the two Siamese Networks a baseline patch and a corresponding mass or calcification patch. The Siamese Networks give in output 1 representation each. These two representations are then concatenated and given as input to 1 fully connected layer that give as result 0 if the input tuple is made by a baseline and a mass patch or 1 if it is made by a baseline and a calcification patch.

Layer (type)	Output Shape	Param #	Connected to
Baseline (InputLayer)	[(None, 150, 150, 1)	0	
Abnormality (InputLayer)	[(None, 150, 150, 1)	0	
Siamese (Functional)	(None, 512)	3452032	Baseline[0][0] Abnormality[0][0]
Concatenation (Concatenate)	(None, 1024)	0	Siamese[0][0] Siamese[1][0]
dropout (Dropout)	(None, 1024)	0	Concatenation[0][0]
dense_24 (Dense)	(None, 512)	524800	dropout[0][0]
dense_25 (Dense)	(None, 1)	513	dense_24[0][0]

Total params: 3,977,345
Trainable params: 3,977,345
Non-trainable params: 0

In this case we have not been able to solve the overfitting simply with the data augmentation, for this reason we have inserted a dropout layer with rate 0.4. The best combination was the one with dropout and rotation: this is in line with some papers, as they argue that horizontal flipping can negatively affect performances.

	Data Augmentation	Dropout	Epochs	Accuracy	Loss	AUC
SIAMESE_1	Flip + Rotation	☐	50	0,83	0,38	0,83
SIAMESE_2	Flip + Rotation	☐	50	0,82	0,40	0,82
SIAMESE_3	Flip + Rotation	☐	50	0,75	0,51	0,73
SIAMESE_4	Rotation	☐	30	0,82	0,40	0,81
SIAMESE_5	Rotation	☑	50	0,84	0,36	0,83



7 Task 5: Ensemble Classifier

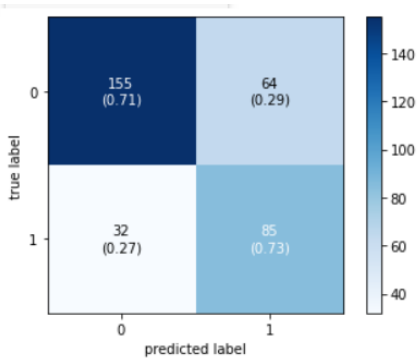
For the Ensemble task we decided to use the Benignant vs Malignant models obtained with the previous tasks hoping to obtain better performances than those we obtained previously. We decided to use:

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

We predicted the test set labels using the 3 models and the results where aggregated together using the average voting. We then computed the metrics comparing the labels obtained from the averages and the true labels. From the tables we can see that even if the Benignant vs Malignant models are not performative, the ensemble can manage to slightly improve the metrics.

	Accuracy	F2	AUC
Best_BM_Scratch	0,65	0,61	0,65
Best_BM_VGG16	0,64	0,68	0,67
Best_BM_Inceptionv3	0,69	0,69	0,7
Ensemble	0,71	0,68	0,71

In the confusion matrix we can see how the False Negatives(32) are kept low by the ensemble, maintaining the f2 with an acceptable value with respect to the overall performances we faced with this dataset.



8 Conclusions

From the analysis carried out and the results obtained, we were able to state that the data set is prepared for the mass-calcification classification problem, doing more tests and trying different combinations of hyperparameters is maybe possible to obtain even better results. On the other hand, we had had a lot of difficulty in developing a satisfactory model for the benign-malignant problem. We have noticed and verified with our tests that, in this case, the images are classified better when a small contrast is applied: we empirically set the contrast factor to 1.5 or to 2. We tried to select the best trade-off between accuracy and f2 values, because is some cases to higher accuracy values corresponded an unacceptable value of right predictions.
Some of our tests, especially in the initial phase, were affected by some problems with the colab platform. In particular, during training, the validation remained fixed at 0.50. This could also happen with a configuration that until the previous execution had produced good results.

9 References

- [1]Code used for concatenating the siamese networks: https://github.com/aspamers/siamese/blob/master/mnist_siamese_example.py
- [2]Code to build a Siamese network: https://www.pyimagesearch.com/2020/11/30/siamese-networks-with-keras-tensorflow-and-deep-learning/
- [3]F2 score in medical imaging issues: https://deepai.org/machine-learning-glossary-and-terms/f-score
- [4]Automatic mass detection in mammograms using deep convolutional neural networks
- [5]Breast Cancer Detection using Deep CNN and SVM
- [6]Deep Learning to Improve Breast Cancer Detection on Screening Mammography
- [7]Transfer Learning and Fine Tuning in Breast Mammogram Abnormalities
- [8]Transfer Learning in Breast Mammogram Abnormalities Classification With Mobilenet and Nasnet