PISA UNIVERSITY

COMPUTATIONAL INTELLIGENCE AND DEEP LEARNING

CONVOLUTIONAL NEURAL NETWORK FOR MEDICAL IMAGING ANALYSIS ABNORMALITY DETECTION IN MAMMOGRAPHY

ACADEMIC YEAR 2020-2021

STEFANO PETROCCHI

SUMMARY

TASK 1 - LITERATURE ANALYSIS

INTRODUCTION

The problem of mammograms classification using ***CBIS DDSM*** or related datasets is very discussed in literature. *Ad-Hoc* convolutional networks for this type of problem are rarely adopted and always bound to well-known *ConvNet* modifications. Thus the main solution that most papers undertake is to use pretrained networks [1] [2].

The most common *ConvNet* used for transfer learning is **AlexNet** for historical reasons. The other three are, in order: **Vgg16**, **Inception** and **ResNet.** Those networks are also the ones with the best results and consequently described in **Task 3** of this project.

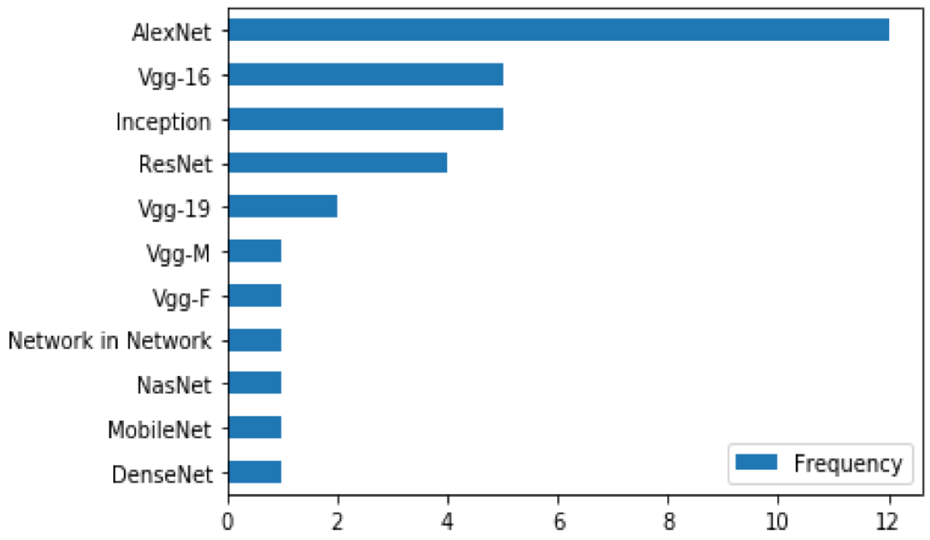


Figure 1: *ConvNets* used in Literature [1]

In mammogram classification there are two main approaches found in literature:

- Processing the *whole mammogram* image [3].
- Processing the region of interest (*ROI*) [4].

The former aim is to find an "end to end design". The approach used in this project is the latter, where the ROI image is extracted (that will be called ***patch*** from now on). In fact processing a whole mammogram images in their original size seems to be a problem itself because mammogram images far exceed the traditional size used in many trained *ConvNets* in the *ImageNet* dataset.

It is essential to note that in the literature patches size is generally superior to $300x300\ p$ [4] [5]. The use of patches less than half in size ($150x150\ p$) and the consequent loss of detail, could be one of the fundamental factors that led to lower results in this project compared to literature in *benign/malignant* abnormality classification, where detail is fundamental to diagnosis. Another observation that confirms this thesis is that the classification between *masses* and *calcifications* turns out to be well in line with the results of the literature, in fact the two classes have macroscopic differences, evident even at low resolution.

Ensemble methods and *Siamese Networks* solutions are discussed directly in ***Task 4*** and ***Task 5*** of this project.

TRANSFER LEARNING & FINE TUNING

TRANSFER LEARNING AS A FEATURE EXTRACTOR

In this case, the pre-trained *ConvNet* is used to extract a feature vector which is later used to train another kind of classifier algorithm like *Support Vector Machines* (**SVM**). This case is illustrated in [5]; the author extracts a feature vector from the last fully connected layers of the pre-trained *AlexNet* and trains *Support Vector Machines* (SVM) on it.

The features extraction can take place on different layers. The method proposed in this project is to flatten the last convolutional layer and connecting it to a SVM classifier, which is the one with the best results in literature for this problem. This model was chosen by empirically comparing its results with those obtained using instead a *global average pooling* or using another layer of the network.

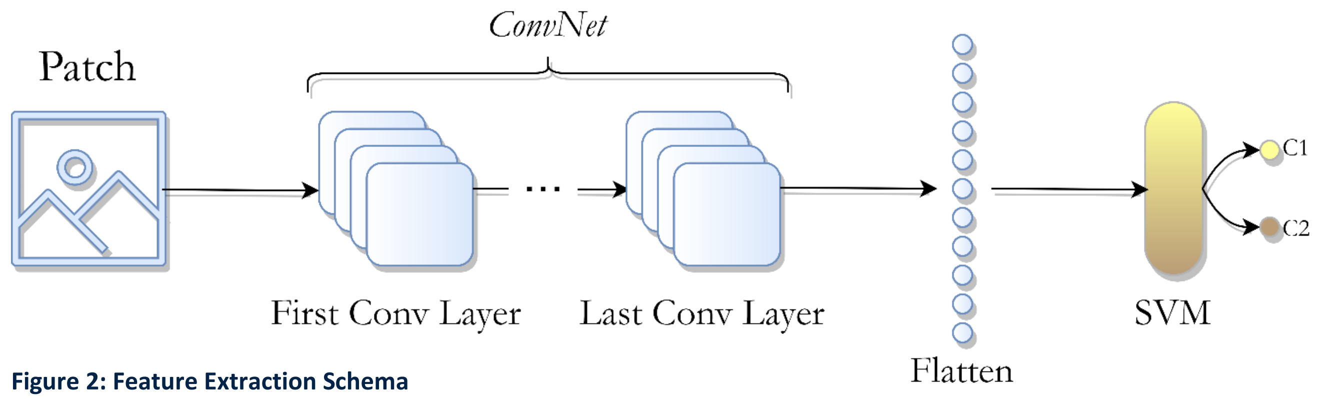


Figure 2: Feature Extraction Schema

TRANSFER LEARNING AS A NEW CONVNET CLASSIFIER

In this case, the last full connecting layer of the *pre-trained ConvNet* may be substituted with a set of additional layers, where the last full connecting layer has only one neuron and the logistic regression for binary classification, or just the number of random initialized neurons required in proportion to the new classification task. For instance, in the case of *benign vs malignant* classification of the mammogram abnormality this can be achieved with a single neuron or two. Only the added layers are trained while the rest of the *ConvNet*'s weights remain frozen.

In this project this methodology was tested in ***Task 3*** and explained in detail in the appropriate section.

TRANSFER LEARNING AS WEIGHT INITIALIZATION

In this case the whole *ConvNet* is *re-train* but uses the values of the ***ImageNet*** pre-trained ConvNet model as initial values for the weights. The last full connecting layer with 1000 categories is substituted by one or two neurons to address the binary classification problem.

FINE TUNING

This is the most common technique found in literature. In this case, the model's last full connecting layer is substituted with the number of neurons needed for the new classification or a set of new layers are added before the output layer. Differently to *transfer learning as a new ConvNet*, some of the last layers of the model are retrained with the new data. For example, *VGG16*, *InceptionV3*, and *ResNet50* are fine tuned in [4]; the author found that when the number of convolutional blocks exceeds 2, the accuracy of the fine-tuned model drops.

PREPROCESSING AND AUGMENTATION

In literature there is some discussion about the impact of both *data augmentation* and *pre-processing* of the medical image. Here are some considerations in this regard:

PREPROCESSING

CLAHE EQUALIZATION

One of the most used techniques is ***CLAHE equalization*** [1] [6] that is computer image processing technique used to improve contrast in images. It differs from ordinary histogram equalization in the respect that the adaptive method computes several histograms, each corresponding to a distinct section of the image, and uses them to redistribute the lightness values of the image. It is therefore suitable for improving the local contrast and enhancing the definitions of edges in each region of an image**.** In this project *CLAHE equalization* has been tested directly on the patches with unsatisfactory results, in most papers it is in fact used before extracting the patch from the original mammography in order to highlight what are the contours of abnormalities, essential to distinguish a benign abnormality from a malignant. Unfortunately, it was therefore not possible in this project to fully test its effectiveness, which could have led in an increase in accuracy of the distinction between benign or malignant abnormalities.

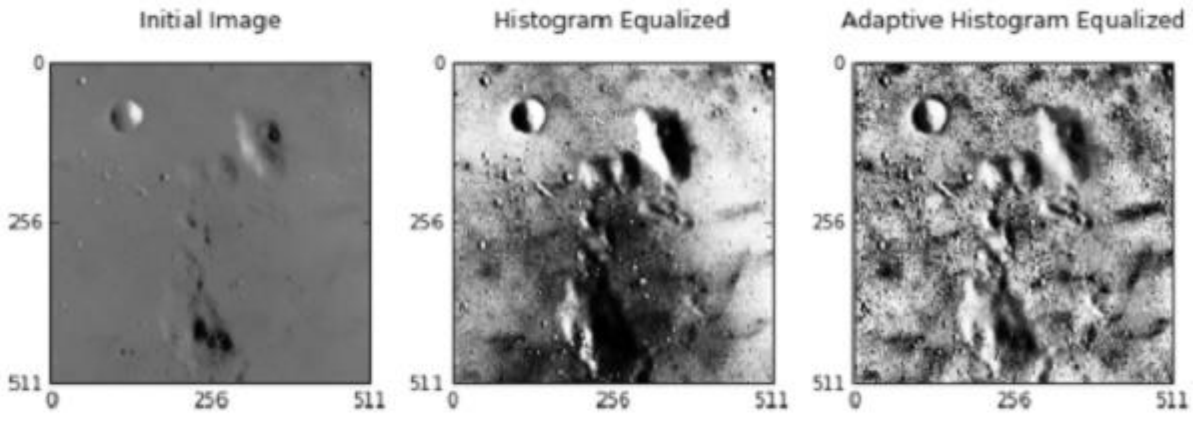


Figure 3: CLAHE and Histogram Equalization Comparison

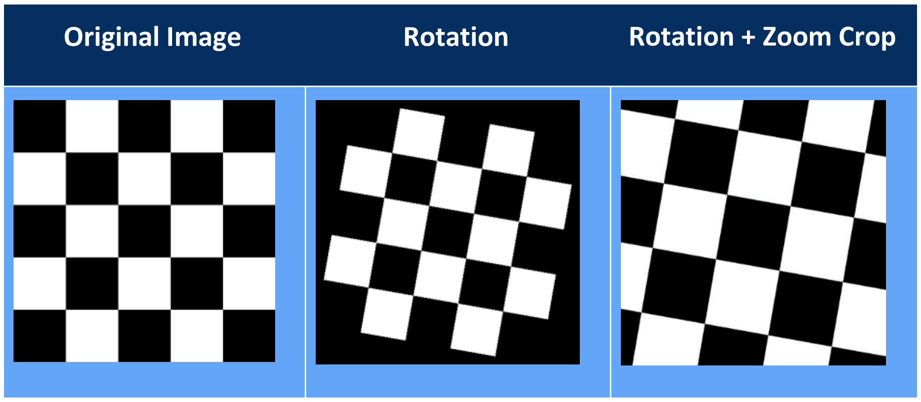
IMAGE NORMALIZATION

It is universally established that the **normalization** of the image allows to obtain better performance in training thanks to a more regular weight evolution. Is in fact essential for each feature to have a similar range so that the gradients don't go out of control. The most commonly used method to perform this type of normalization is to subtract the average value of the *entire training set* from the value of each pixel in the image and to divide the result by its standard deviation, so as to obtain a medium-null and variance one intensity values distribution after normalization.

AUGMENTATION

Since datasets are not so large, data **augmentation** is used by almost all researchers [1] [3] [4] [5]. Some of the most common techniques used are *rotations* and *cropping*. However, the rotation operation yields to distortion of the original image. Because of this reason, *right angle rotations* are preferred in some papers to *random rotation angles,* but this also reduces the effectiveness of improving the network's generalization capabilities.

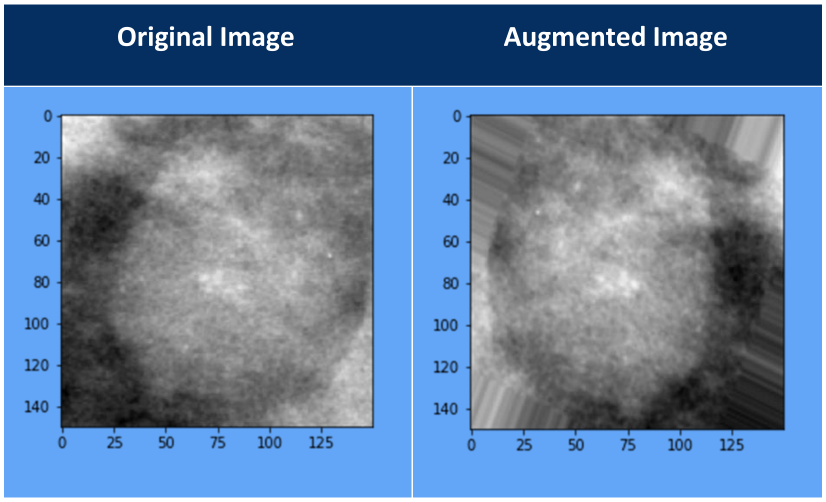
To resolve this issue, it's possible to use libraries such as ***Augmentor Library*** that allows to rotate the image without artificially filling-in the gaps left over, but by zooming-in the image appropriately. Unfortunately, *Augmentor Library* is still in development's early stages and does not fit well with the functions offered by *Keras.*



Therefore, in this project have been tested different types of data augmentation parameters taken from literature using the functions directly offered by *Keras*. As described above, the use of overly invasive image alterations such as random *brightness or intensity variation* or *image deformations* do not allow *ConvNet*s to properly learn the problem, causing *underfitting*.

Empirically the best results were achieved using the parameters suggested by paper [4] and shown below:

- **Horizontal Flip**
- **Rotation Range**: 30°
- **Zoom Range**: [0.75, 1.25]
- **Fill**: "Nearest"



The results confirm that the use of *augmentation* significantly improves the accuracy of large networks (more extensive experiments are described in **Task 2**) and that it is preferable to use parameters that do not distort the image, but it is also necessary to use *random rotations* and not only *right-angle rotations* for these effects to occur.

It is worth mentioning that the use of the *"rotation + zoom crop"* technique could lead to significant benefits if used instead of the *"nearest fill"* that *Keras* functions force to use.

TRAINING

The parameters used in training, as highlighted in some papers, are not a secondary aspect to reach good results.

EARLY STOPPING

Although in some research *CNNs* are trained for *a priori* number of *epochs* [3], the use of **early stopping** allows to make sure to reach an optimum without running the risk of *overfitting* [4]. The features offered by *Keras* functions allows to set very high patience values so as not to stop too early in training. At training's end *CNN's* wights are restored to those values that obtained the best result in validation. This technique has therefore always been adopted in the following project tasks.

LOSS FUNCTION

The *loss function* in this project is set to ***binary cross entropy*** and was chosen because the classification layer used is a *Sigmoid* activation function for all models to discriminate between two classes using one neuron. In literature there are no significant impacts in different choice of this hyperparameter.

OPTIMIZATION ALGORITHM [7]

The next choice to make was that of the type of *optimizer* to use. ***RMSProp***, an adaptive gradient algorithm, is frequently used in computer vision tasks. In many papers is indicated how *RMSProp* outperforms other common optimization algorithms like *Stochastic Gradient Descent (**SGD**)* that maintains a single learning rate during training.

In the literature, however, it is possible to notice that some of the best results obtained in classification of mammograms are achieved using ***Adam*** an extension to *stochastic gradient descent* that has recently seen broader adoption for deep learning applications in computer vision and natural language processing. *Adam* realizes the benefits of both ***AdaGrad*** and *RMSProp* and instead of adapting the parameter learning rates based on the average first moment (the mean) as in *RMSProp*, *Adam* also makes use of the average of the second moments of the gradients (the uncentered variance). The results obtained empirically in this project confirm that *Adam* allows to increase the accuracy of the best models by *1-3%*.

LEARNING RATE

In the literature, values ranging from 10^{-3} to 10^{-5} are typically used as learning rates. *Adam* being an adaptive algorithm decreases this value in advanced stages of training independently. However, it has been empirically tested in this project that the use of a low learning rate ($\mathbf{10^{-5}}$) allows to have a much more regular trend in training, with respect to the use of a higher value. This is fundamental to avoid obtaining oscillating values in the final stages of the training process. These results can be seen in the subsequent task's graphs.

BATCH SIZE [8]

The *batch size* choice is a very controversial topic in literature and is also strongly related to the *learning rate* and *optimization algorithm* used.

A large batch allows computational speedups from the parallelism of *GPUs*. At the same time using a batch equal to the entire dataset guarantees convergence to the global optima of the objective function. However, this is at the cost of slower, empirical convergence to that optima and sometime cause *overfitting*.

Using small batches is not a choice if enough *memory* for larger batches is not available. In addition, using fewer examples results in a less accurate estimate of the error gradient that is highly dependent on the specific training examples used. This results in a noisy estimate that, in turn, results in noisy updates to the model weights, e.g. many updates with perhaps quite different estimates of the error gradient. Nevertheless, these noisy updates can result in faster learning and sometimes a more robust model.

Taking these aspects into account, the value typically used for batch size ranges from *32* (default) to *few hundred* samples. In this project it has been empirically tested that a value of ***107*** for batch size (which results in 20 exact batches per epoch using the dataset available) is an excellent value that allows to obtain an good performance for training when used in conjunction with the other parameters described above.

TASK 2 – SCRATCH CNNS

EMPIRICAL CONSIDERATIONS

Here are some empirical considerations that have been taken into consideration for creating the final "from scratch" models.

DROPOUT

The advantages that the ***dropout*** technique brings in avoiding the phenomenon of overfitting are well known in literature, therefore its use was also considered in this project. By empirically testing models with and without dropout, it was confirmed that it is essential to avoid the phenomenon of *overfitting* in each model used, moreover in conjunction with a low learning rate it is sufficient even for the most complex models.

L1-L2 REGULARIZATION

Regularization is used to prevent network weights from taking high values that contribute to the phenomenon of *overfitting* and *exploding gradient*. In this project, empirical tests were carried out to determine what type of regularization was the most suitable and with what values. Regularizations ***l1*** and ***l2*** were tested and in both cases they didn’t significantly contribute to increase training performance, in many cases they instead caused *underfitting*. Therefore, it was not used in the creation of the final models.

BATCH NORMALIZATION [9]

Batch normalization is a method used to make networks faster and more stable through normalization of the input layer by re-centering and re-scaling. While the effect of batch normalization is evident in many papers, the reasons behind its effectiveness remain under discussion. Several batch normalization configurations were tested (is debated even where to insert it), these increased the speed of the training process, but their use also resulted in faster *overfitting* and worse model quality. Therefore, it was not included in the final models.

FULLY CONNECTED LAYERS

Two fully connected layers are the standard used in most "pre-trained" networks for classification purposes, but in this project a *single* layer was also tested in order to reduce the complexity of the networks. Different values were also tested for the number of *neurons* per layer, between: {*128, 256, 512, 1024, 2048, 4096*}.

However, different combinations of all these values did not had empirically different results, the only combination that seems to obtain an average of *1-2%* more accuracy (in validation and test) was the use of ***two fully connected layers*** with ***4096*** neurons each.

GLOBAL AVG POOLING VS FLATTEN

Between the *convolutional* layers and the *fully connected* layers of the classifier it’s generally necessary to use one among *Global AVG Pooling* and *Flatten* layers. The first takes the global average for each feature map, allowing to considerably reduce the number of neurons, the second simply reshapes the tensor resulting from the convolution by linearizing it.

The results of the experimental tests carried out in this project have shown that the use of a ***flatten*** layer, although heavier, helps to obtain greater accuracy, if the overfitting phenomenon is managed through *augmentation* and *dropout*.

PADDING AND STRIDE

Padding is commonly used in *ConvNet* design as it allows to maintain the size of the *feature maps* even by applying different convolutions consecutively, this allows to build deeper networks, as not using padding means reducing the size of the *feature maps* at each convolution. Another reason why padding is commonly used is that allows to preserve the information present on the edges of the image, which would otherwise be affected by fewer *local receptive fields* than the central areas of the image.

Despite these advantages, empirically testing the use of padding in this project gave unsatisfactory results, most likely because patch edges are not of particular interest and too deep networks are not effective in this type of problem. Therefore, it wasn't used in the final models.

Stride allows to reduce the size of the output feature spaces, but generally it's better to use *max pooling* instead as stride can lead to information loss.

MODELS

TINY-NET

This network has been designed to be as simple as possible while maintaining good effectiveness. It's made up of 5 ***convolutional*** *3x3* blocks with *2x2* ***max pooling*** for each block. The number of ***kernels*** per convolution is kept low and goes from *16* to *32* to end up with three blocks of *64* filters. A ***flatten*** layer is used after the last pooling. To keep the network simple, a *single* ***fully connected*** layer with *512* neurons is used. a strong ***dropout*** (*0.5*, *0.2*), which seems to have very good effects on the network, has also been included.

```
Model: "tiny_net"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_conv (Conv2D)          (None, 148, 148, 16)      160       
_________________________________________________________________
input_max_pooling (MaxPoolin (None, 74, 74, 16)        0         
_________________________________________________________________
conv_1 (Conv2D)              (None, 72, 72, 32)        4640      
_________________________________________________________________
max_pooling_1 (MaxPooling2D) (None, 36, 36, 32)        0         
_________________________________________________________________
conv_2 (Conv2D)              (None, 34, 34, 64)        18496     
_________________________________________________________________
max_pooling_2 (MaxPooling2D) (None, 17, 17, 64)        0         
_________________________________________________________________
conv_3 (Conv2D)              (None, 15, 15, 64)        36928     
_________________________________________________________________
max_pooling_3 (MaxPooling2D) (None, 7, 7, 64)          0         
_________________________________________________________________
conv_4 (Conv2D)              (None, 5, 5, 64)          36928     
```

```
_________________________________________________________________
max_pooling_4 (MaxPooling2D) (None, 2, 2, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 256)               0
_________________________________________________________________
drop1 (Dropout)              (None, 256)               0
_________________________________________________________________
fully_connected (Dense)      (None, 512)               131584
_________________________________________________________________
drop2 (Dropout)              (None, 512)               0
_________________________________________________________________
fully_connected_out (Dense)  (None, 1)                 513
=================================================================
Total params: 229,249
Trainable params: 229,249
Non-trainable params: 0

_________________________________________________________________
```

MAMMO-NET

This network has been designed to achieve the maximum possible performance and is inspired by ***VGG16***, that appears to be the best network for this type of classification. The network consists of *3x3* ***convolutions*** and *2x2* ***max pooling*** divided into 4 blocks. The first two blocks contain two *convolutions* and one *pooling* each, the last two three *convolutions* and two *pooling* each. An increasing number of ***kernels*** were used (from *16* to *128*), but fewer than in *VGG16* as more *kernels* caused *overfitting*. Another difference with *VGG16* is that ***padding*** was not used for the reasons described in precedence. The classification layers are composed of a ***flatten*** layer and two ***fully connected*** layers with *4096* neurons, all these measures have been explained previously. Moderate dropouts (*0.2, 0.2, 0.1*) were used to reduce *overfitting*.

```
Model: "mammo_net"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_conv1 (Conv2D)         (None, 148, 148, 16)      160
_________________________________________________________________
input_conv2 (Conv2D)         (None, 146, 146, 16)      2320
_________________________________________________________________
input_max_pooling (MaxPoolin (None, 73, 73, 16)        0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 71, 71, 32)        4640
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 69, 69, 32)        9248
_________________________________________________________________
block1_max_pooling (MaxPooli (None, 34, 34, 32)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 32, 32, 64)        18496
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 30, 30, 64)        36928
_________________________________________________________________
block2_conv3 (Conv2D)        (None, 28, 28, 64)        36928
_________________________________________________________________
block2_max_pooling (MaxPooli (None, 14, 14, 64)        0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 12, 12, 128)       73856
_________________________________________________________________
```

```
block3_conv2 (Conv2D)        (None, 10, 10, 128)       147584    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 8, 8, 128)         147584    
_________________________________________________________________
block3_max_pooling (MaxPooli (None, 4, 4, 128)         0         
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0         
_________________________________________________________________
drop1 (Dropout)              (None, 2048)              0         
_________________________________________________________________
fully_connected_1 (Dense)    (None, 4096)              8392704   
_________________________________________________________________
drop2 (Dropout)              (None, 4096)              0         
_________________________________________________________________
fully_connected_2 (Dense)    (None, 4096)              16781312  
_________________________________________________________________
drop3 (Dropout)              (None, 4096)              0         
_________________________________________________________________
fully_connected_out (Dense)  (None, 1)                 4097      
=================================================================
Total params: 25,655,857
Trainable params: 25,655,857
Non-trainable params: 0

_________________________________________________________________
```

TRAINING AND RESULTS

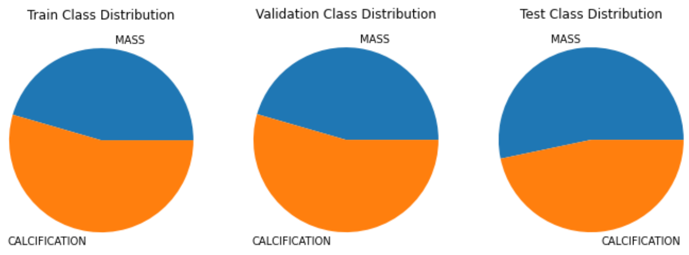
INTRODUCTION TO THE EXPERIMENTS

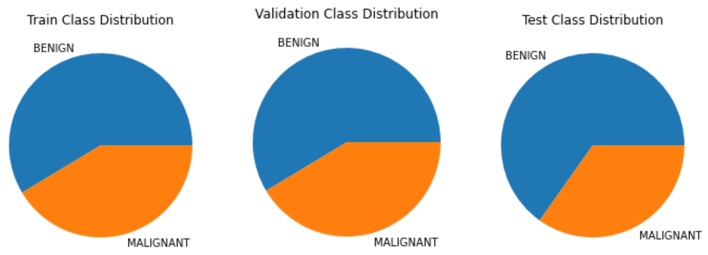
The two networks created *from scratch* were trained both to distinguish between ***masses and calcifications*** and between ***benign and malignant*** abnormalities. For each network and each type of classification, the training was performed both ***with augmentation*** and ***without augmentation*** to test its effectiveness. Furthermore, *feature extraction* was performed for each model produced in order to train ***SVM classifiers*** on the extracted features and evaluate their effectiveness.

The ***parameters*** used to train the networks are the following:

- **Loss:** binary cross entropy
- **Optimizer:** Adam
- **Learning rate:** 10^{-5}
- **Batch size:** 107
- **Max epochs:** 1000
- **Patience:** 50

Class distribution of the dataset used:





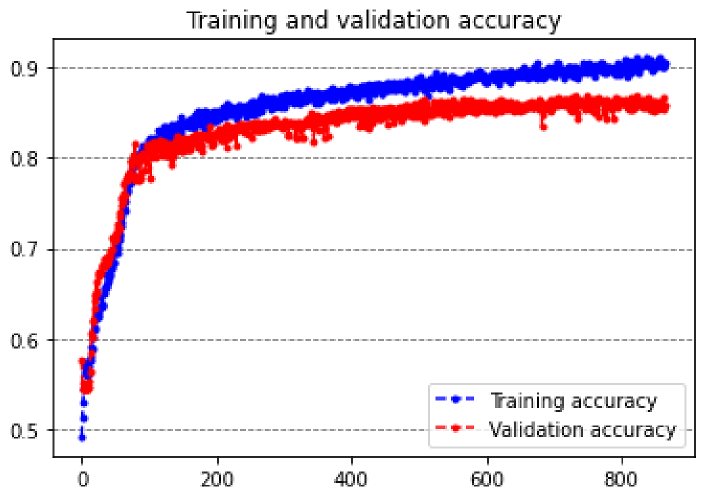
It was used a ***stratified split***, in order to maintain the original balancing, splitting *validation* from the *training* set. The train and validation classes are slightly biased for both type of classification but interventions for their rebalancing do not seem necessary. Unfortunately, the test set turns out to be clearly unbalanced in favor of the *benign* class for the *benign-malignant* classification, but since it is already provided in this way no interventions have been performed for its rebalancing.

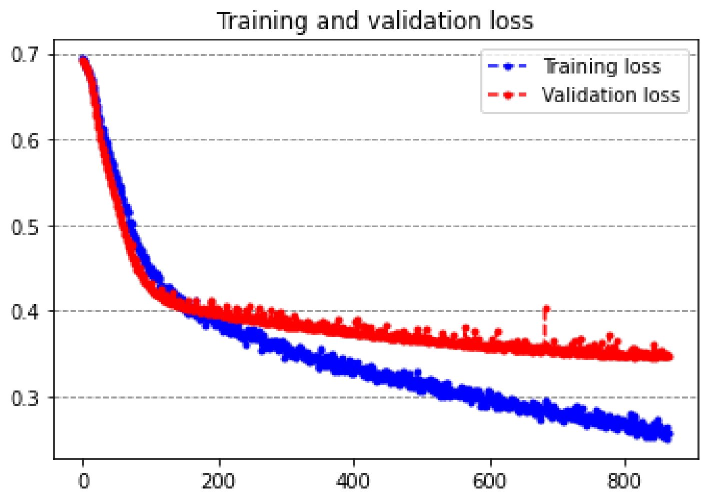
Note: When reading these results, it must be taken into account that the same subdivision has always been used between *training*, *validation* and *test* set for each model and that the values refer to a single model result. To obtain reliable data, the average value and the standard deviation should be considered instead, repeating the experiment several times for each model, which unfortunately was not possible in this project due to the limitations imposed by *Colab*.

MASS-CLACIFICATIONS CLASSIFIERS

TINY-NET

Training Without Augmentation

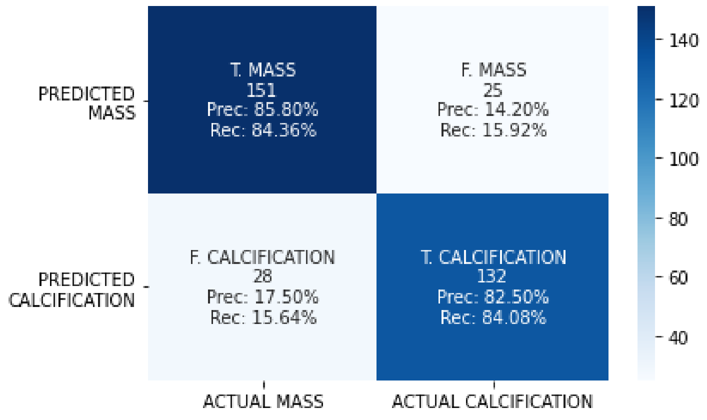




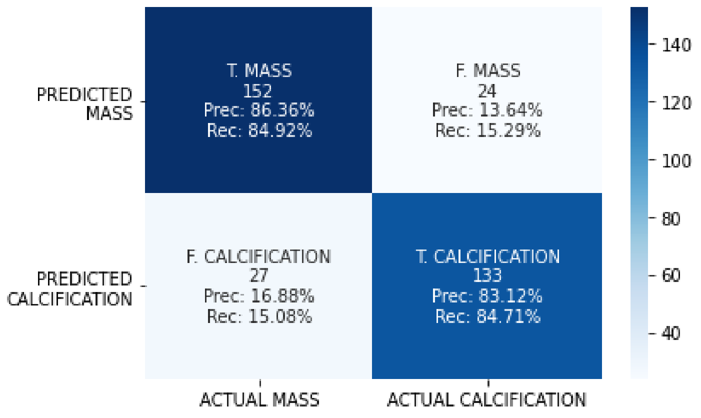
From the progress of the training it can be seen how the use of a *simple network* and a very accentuated *dropout* allow to mitigate the effects of *overfitting* even without the use of *augmentation*. It can also be noted a logarithmic trend of the training, which thanks to the use of a low learning rate is very stable, but also requires a high number of *epochs*.

Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss	SVM Test Accuracy
91.31%	0.2426	86.57%	0.3448	**84.23%**	**0.3553**	**84.82%**

TinyNet Confusion Matrix:

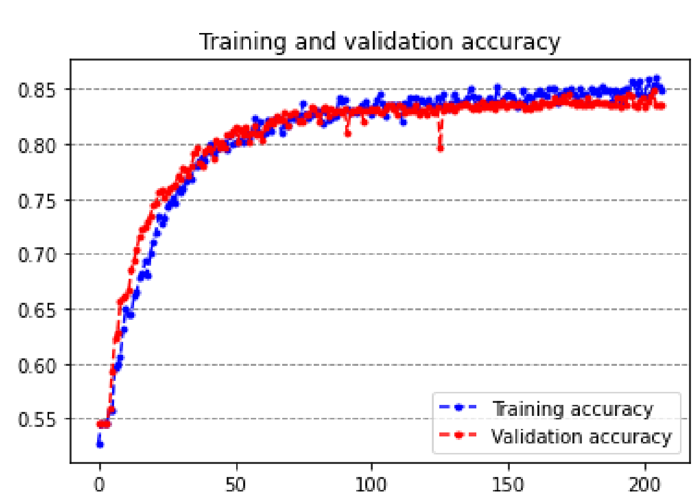


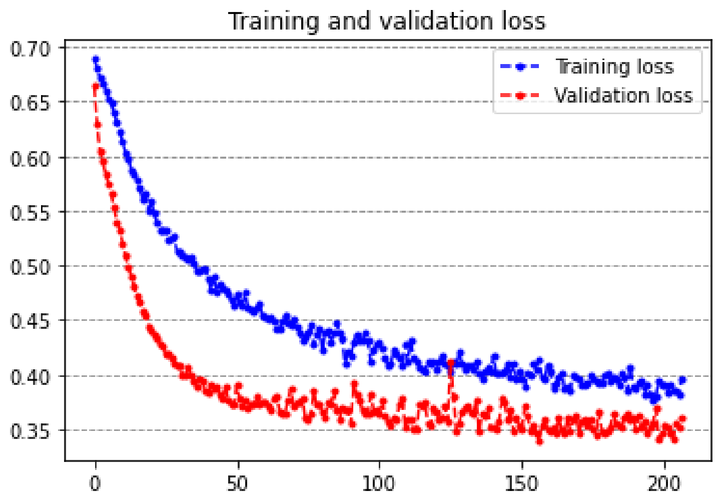
SVM Confusion Matrix:



As can be seen in the confusion matrices, the classification is ***balanced***. Moreover, SVM classification on extracted features does not improve classification.

Training with Augmentation

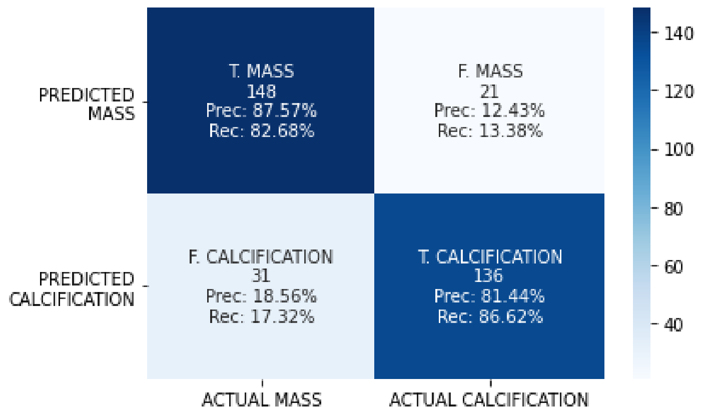




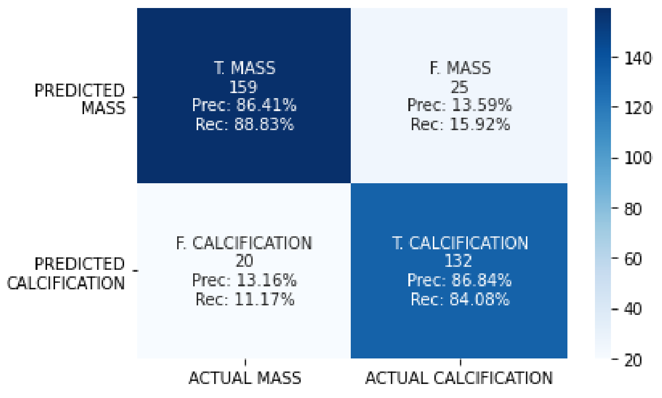
Also in this case the training curve is logarithmic, as expected from an adaptive optimizer like *Adam*. It is interesting to note that the validation has better loss values than the training, this is probably due to the use of *augmentation* since this phenomenon did not occur in the previous experiment. Augmentation makes it more difficult for the classifier to classify training samples, leading to worse accuracy. In validation the accuracy is higher as augmentation is not used and it is easier for the classifier to classify the unaltered images.

Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss	SVM Test Accuracy
84.3%	0.3938	83.21%	0.4066	**84.52%**	**0.3793**	**87.5%**

TinyNet Confusion Matrix:



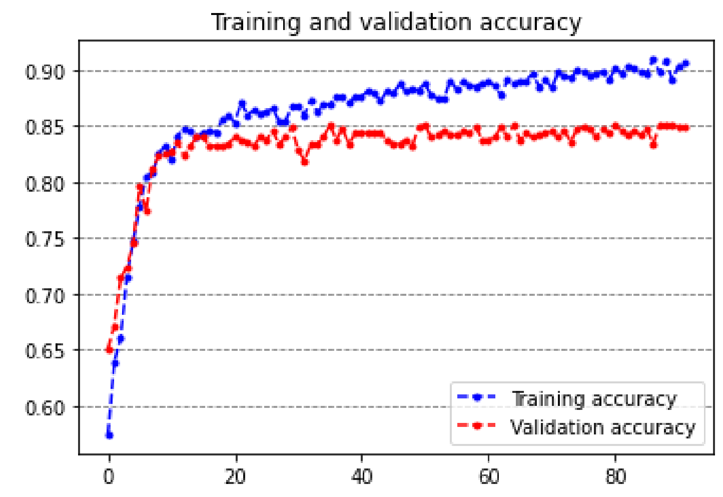
SVM Confusion Matrix:

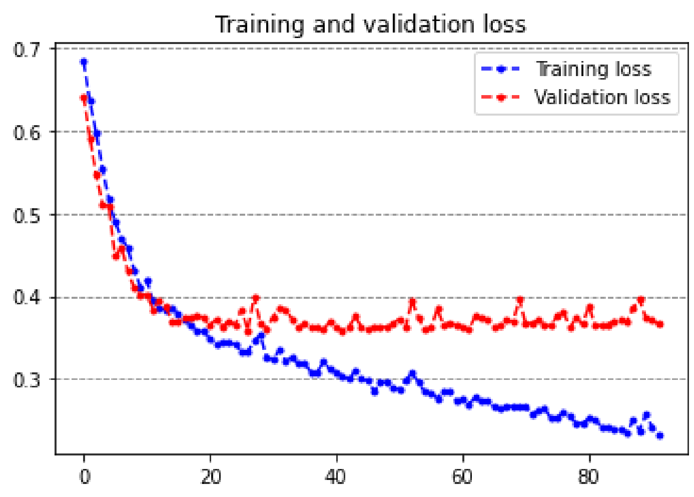


In this experiment it is interesting to note how the ***SVM*** classifier, rebalancing the classification, obtains optimal results on the features extracted using a simple network such as *TinyNet*. This highlights the potential of SVM classifiers in improving non-optimal models starting from their features, unfortunately following the results described in the subsequent experiments it is possible to note that for already good and balanced models it is not helpful.

MAMMO-NET

Training Without Augmentation

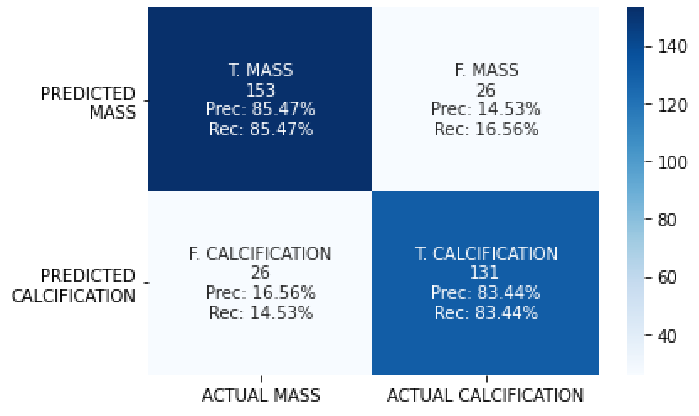




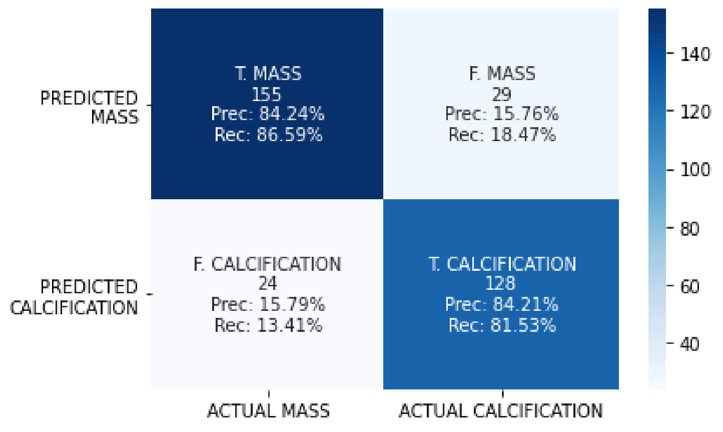
The course of the training is very similar to that of TinyNet without the use of augmentation, the difference is that in this case, given the much larger size of the network, the dropout is not sufficient to counteract the overfitting which therefore immediately explodes.

Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss	SVM Test Accuracy
88.41%	0.2882	84.33%	0.3589	**84.52%**	**0.4109**	**84.23%**

MammoNet Confusion Matrix:

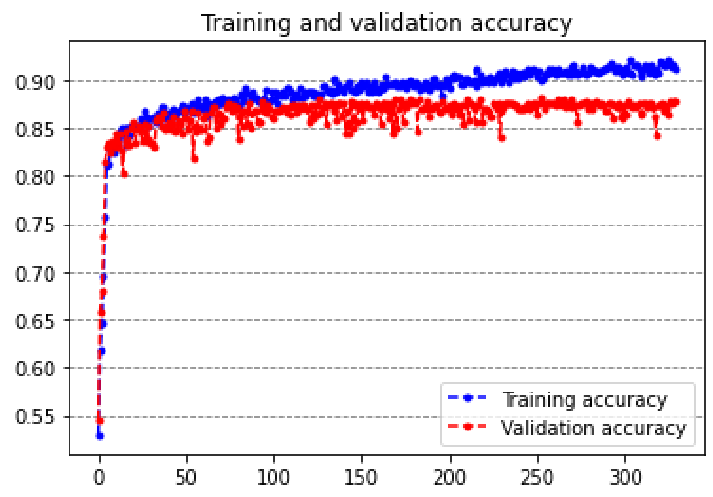


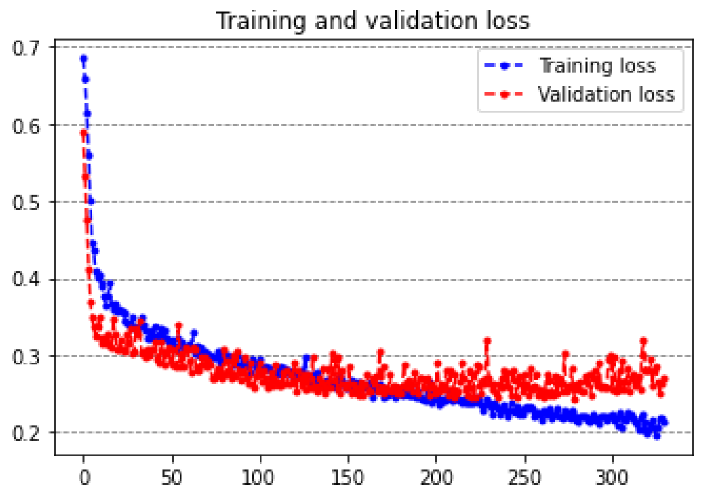
SVM Confusion Matrix:



This is an example where the use of an *SVM classifier* does not lead to improvements over an already balanced classification. It is worth mentioning that despite the overfitting the choice of using the ***early stopping*** always allows to obtain the best possible model given a certain initial configuration.

Training with Augmentation

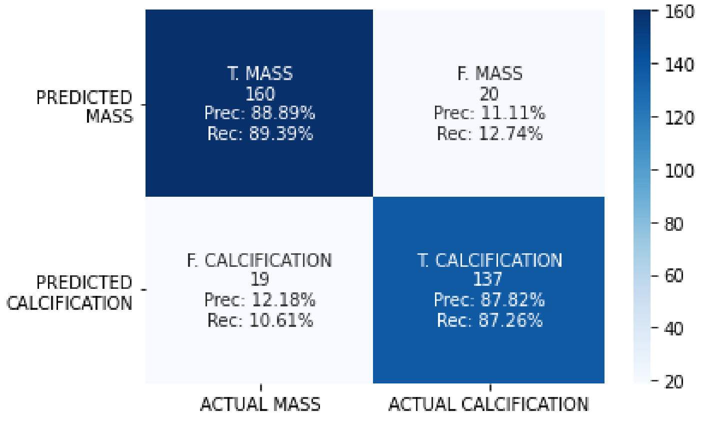




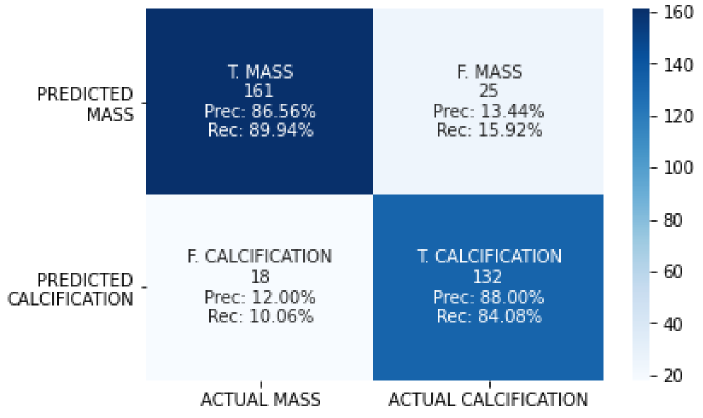
In this case the training turns out to be extremely fast in the first epochs, and then slowly improves the confidence in the classification (reducing the loss). A slight *overfitting* is always present, but the use of ***augmentation*** allows to keep it very low if we consider the size of the network used. *Augmentation is therefore essential for large networks.*

Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss	SVM Test Accuracy
92.48%	0.1865	86.94%	0.289	**88.39%**	**0.3024**	**86.9%**

MammoNet Confusion Matrix:

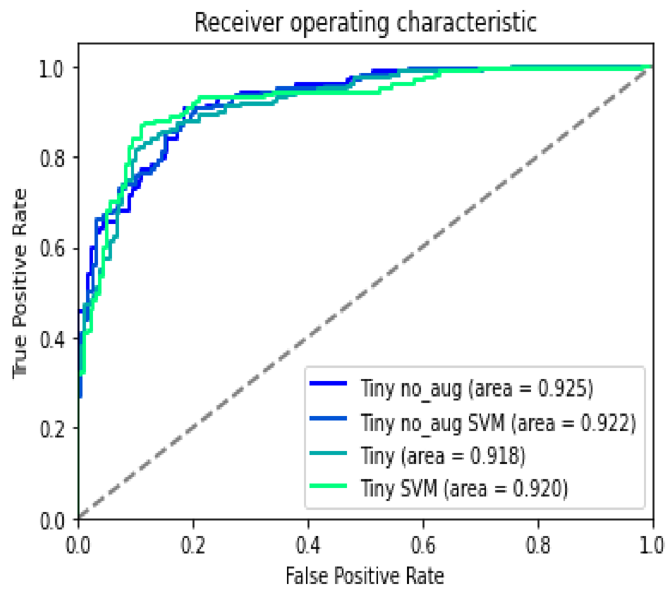


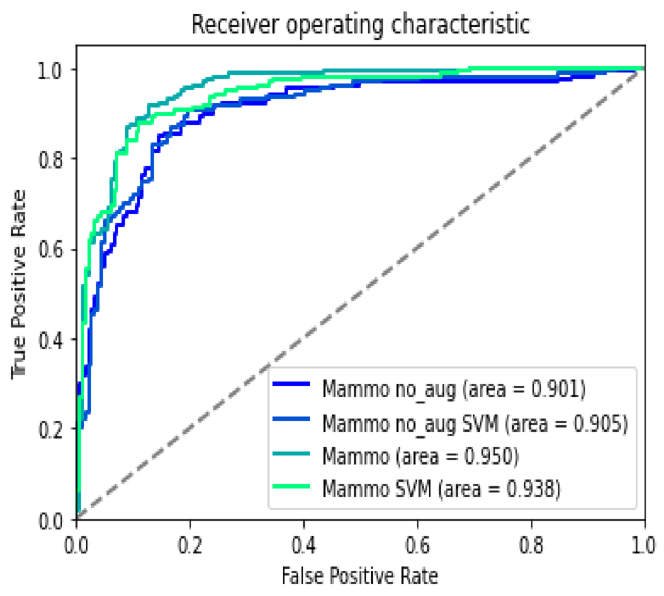
SVM Confusion Matrix:



The classification is excellent and well balanced. The use of a *large network* with *dropout*, *augmentation, low learning rate* and *early stopping* is therefore a great combination. As previously described the use of an *SVM classifier* does not improve the result if the classification is already balanced.

RESULTS





The use of **roc curves** allows to objectively compare binary classifiers, beyond their balance and the thresholds used for the classification. The *area under the roc curve* (***AUC***) can therefore be used as an objective measure of network quality even in unbalanced problems.

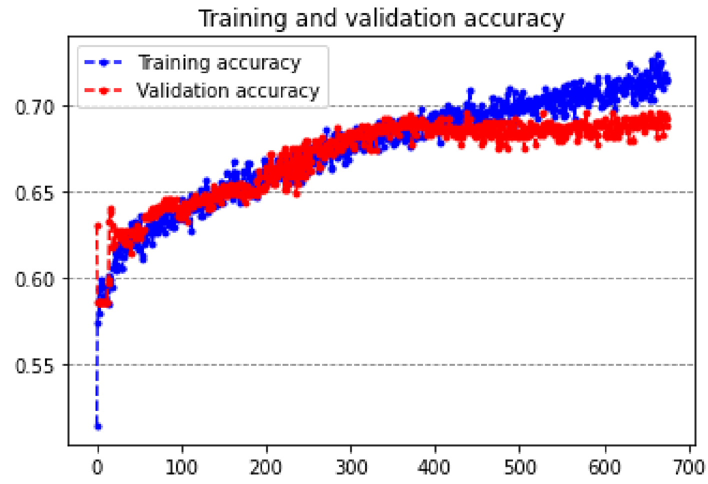
Among the various versions of ***TinyNet*** none is prevalent, the differences may be due to random factors. this highlights how the *SVM classifier* built on the *TinyNet with augmentation*'s features is not structurally better than the others, it only chooses a better threshold to obtain maximum accuracy.

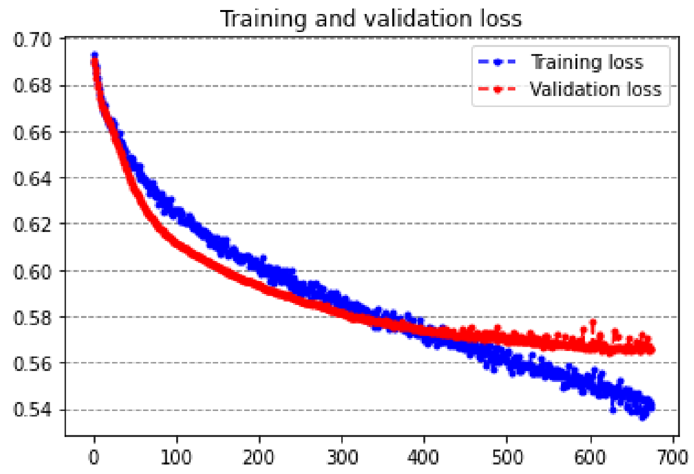
Among the various classifiers based on ***MammoNet*** it is evident how the version that uses *augmentation* is structurally better than the others, having a curve closer to the optimal one in each of its points compared to the others. The results obtained using a *from scratch* network and with *low quality and quantity data*, highlight how this network is excellent for this type of classification.

BENIGN-MALIGNANT CLASSIFIERS

TINY-NET

Training Without Augmentation

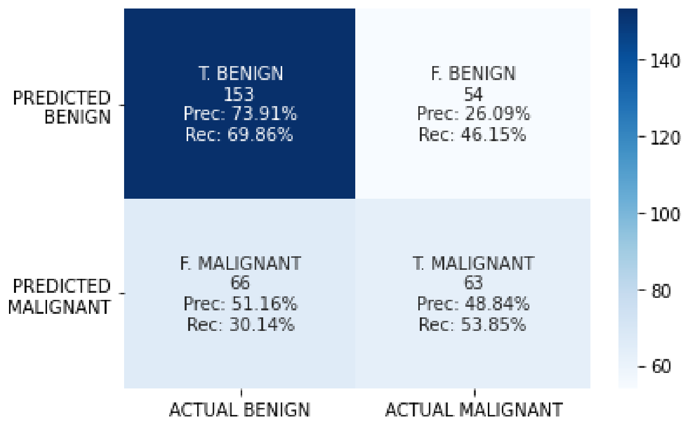




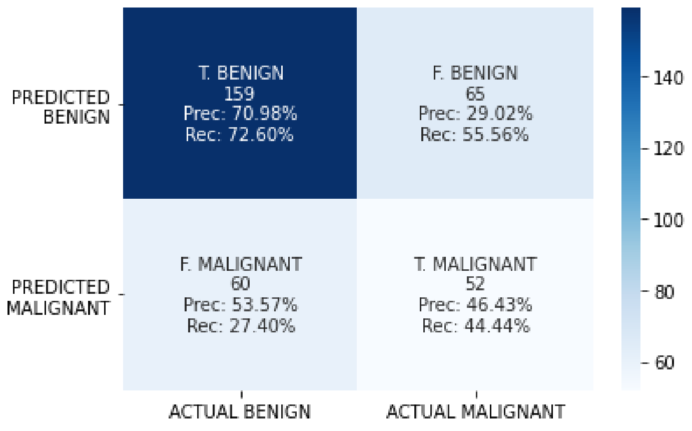
The training regarding the *benign-malignant* classification is not as stable as for the *mass-calcification* classification, but still maintains a logarithmic trend. there is a fair amount of *overfitting*, but only in the most advanced training phases.

Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss	SVM Test Accuracy
71.36%	0.5381	68.84%	0.5645	**64.29%**	**0.6159**	**61.31%**

TinyNet Confusion Matrix:

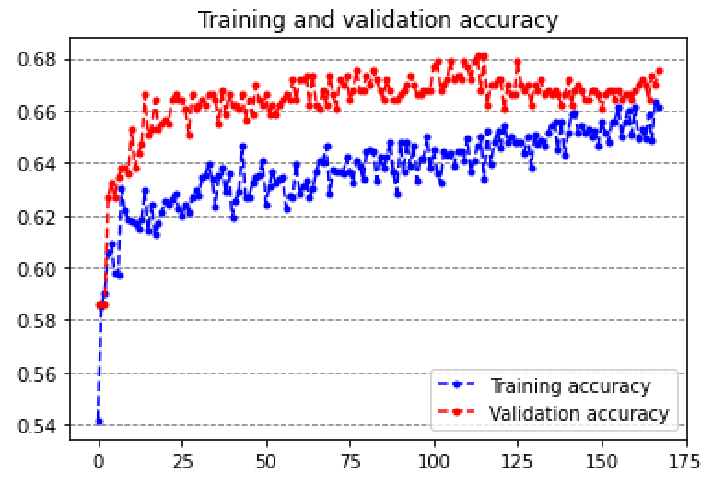


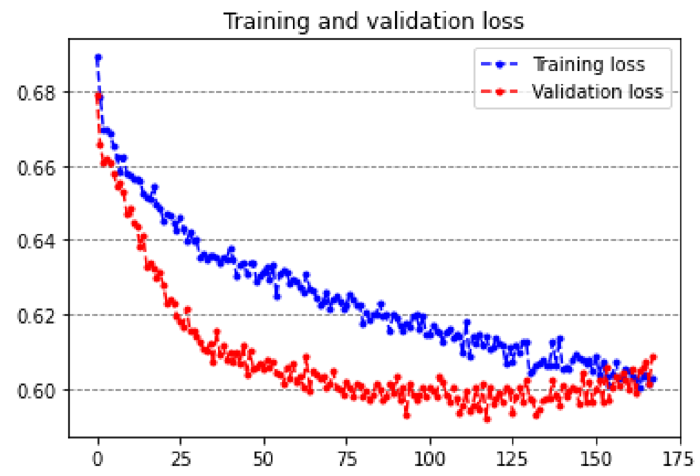
SVM Confusion Matrix:



From the confusion matrices it is evident that the classifier is clearly better in identifying *benign abnormalities*. However, it is not only due to unbalance as the ***precision*** in the classification of benign abnormalities is also high. The classifier is confident when dealing with benign abnormalities and randomly choose when dealing with malign abnormalities. The SVM classifier skews the classification further towards *benign abnormalities*, increasing *recall* but also decreasing *precision* and resulting in lower overall *accuracy*.

Training with Augmentation

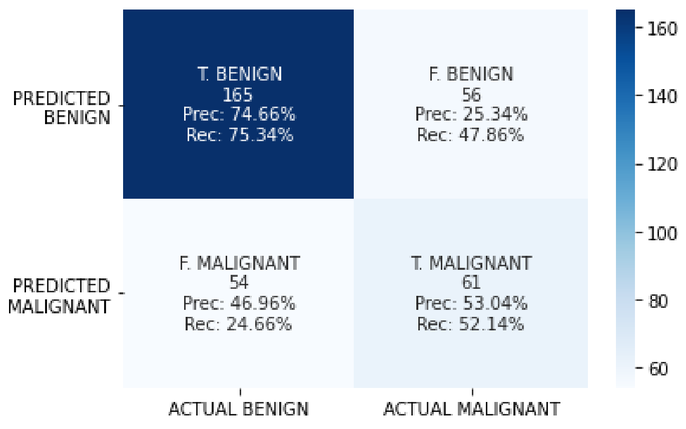




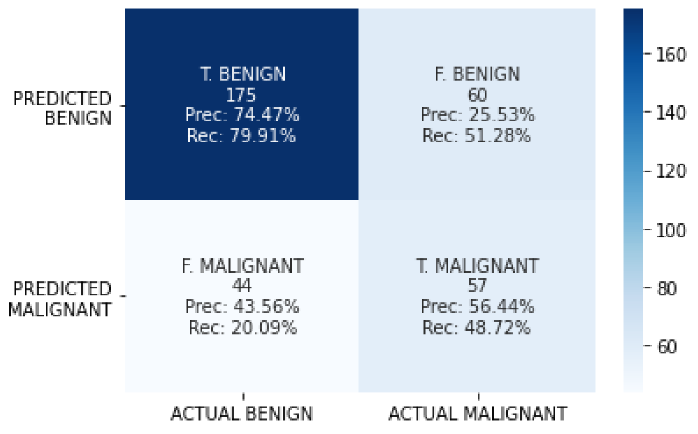
As explained above, the reason why the validation results are better than the training ones is the use of augmentation in training, a theory that is confirmed here.

Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss	SVM Test Accuracy
65.7%	0.6032	66.98%	0.5863	**67.26%**	**0.623**	**69.05%**

TinyNet Confusion Matrix:



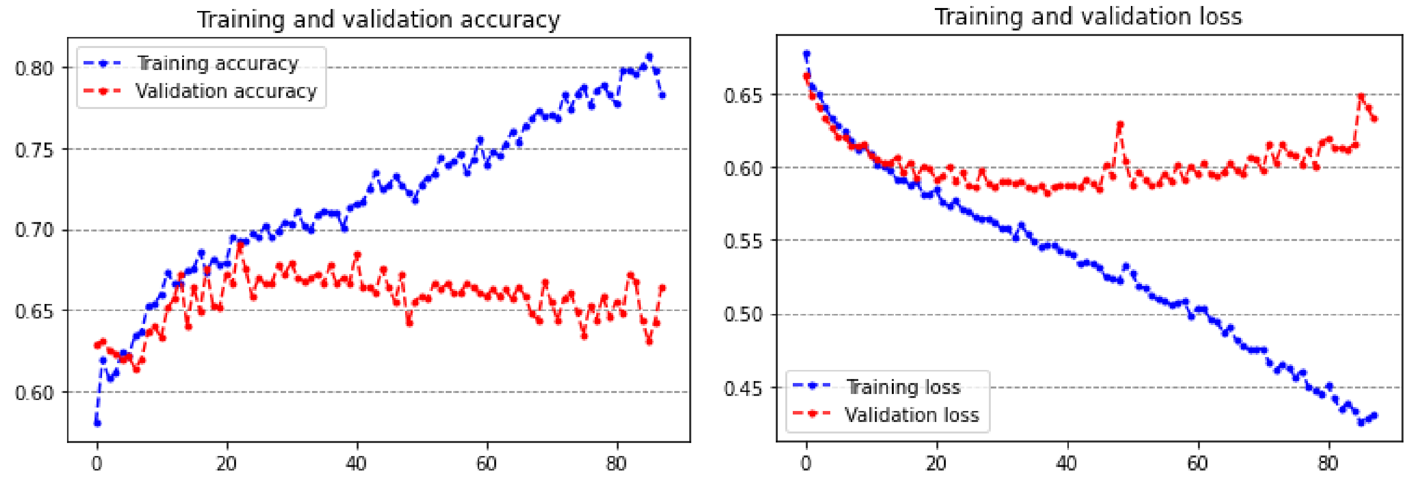
SVM Confusion Matrix:



The same considerations on the balancing of the classifier mentioned above can be applied in this case. Furthermore, also in this case the SVM classifier prefers the classification of *benign abnormalities*. The accuracy is higher as the benign class is the majority one in the test set.

MAMMO-NET

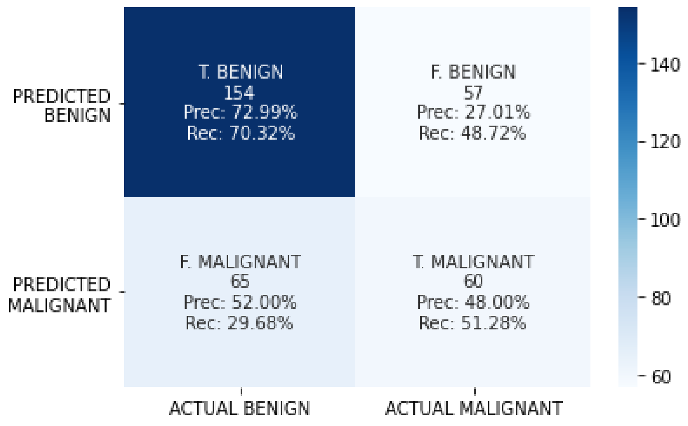
Training Without Augmentation



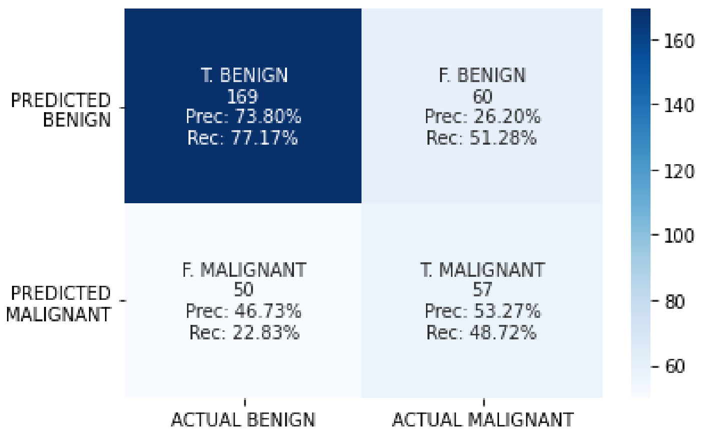
It is possible to notice that for the *benign-malignant* classification ***overfitting*** is a real problem, in fact it occurs from the earliest epoch not allowing to fully exploit the potential of the network

Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss	SVM Test Accuracy
72.01%	0.5353	66.6%	0.5824	**63.69%**	**0.6353**	**67.26%**

TinyNet Confusion Matrix:

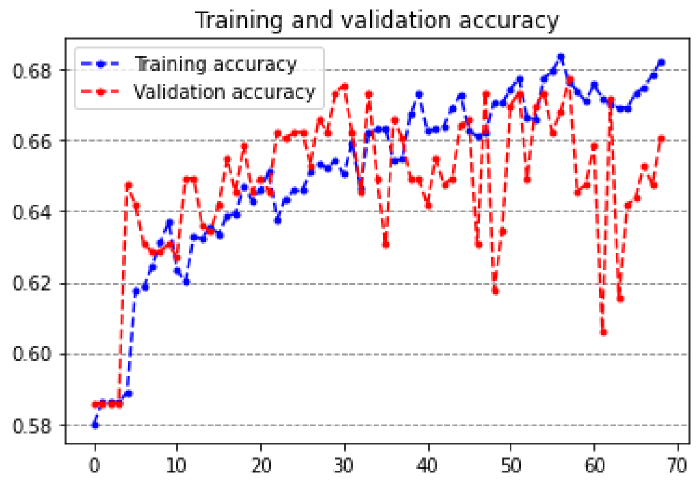


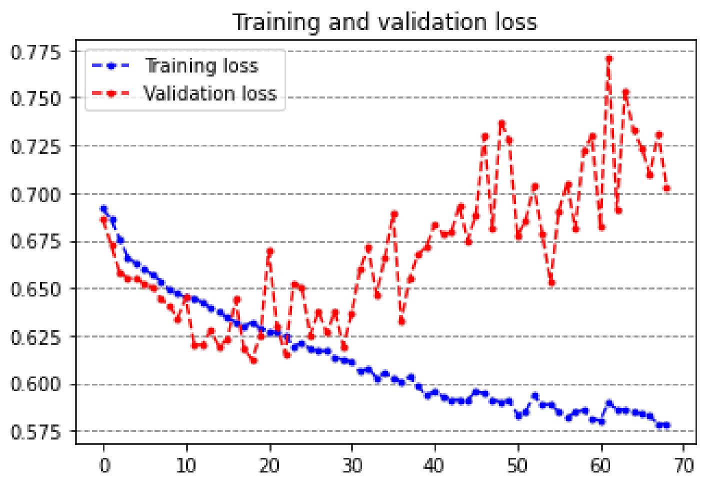
SVM Confusion Matrix:



Again, the balancing of the classification is in favor of *benign abnormalities*, which seems to be an intrinsic factor in the problem. Furthermore, even here the SVM classifier is biased towards benign abnormalities classification.

Training with Augmentation

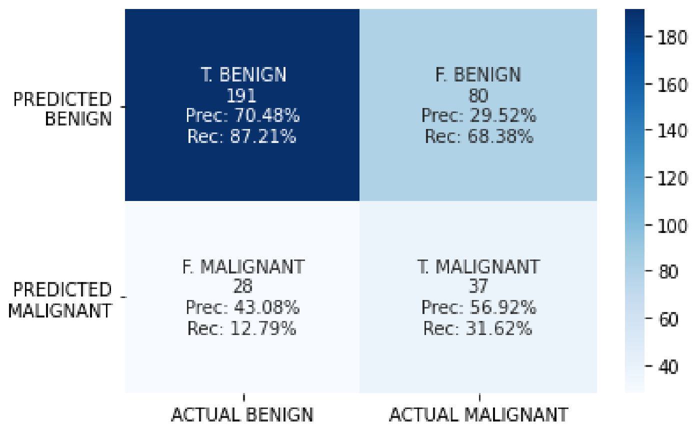




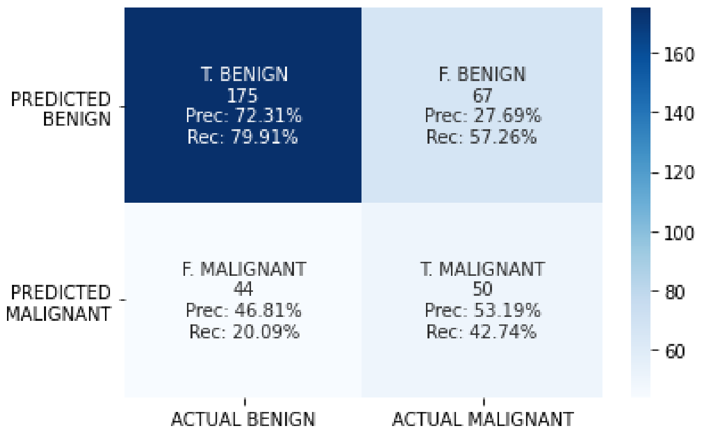
Also in this case the graph shows a strong *overfitting*, which however seems to be more contained, even if the training is concluded earlier than the previous one.

Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss	SVM Test Accuracy
63.88%	0.6337	65.86%	0.6242	**67.86%**	**0.6154**	**67.26%**

TinyNet Confusion Matrix:

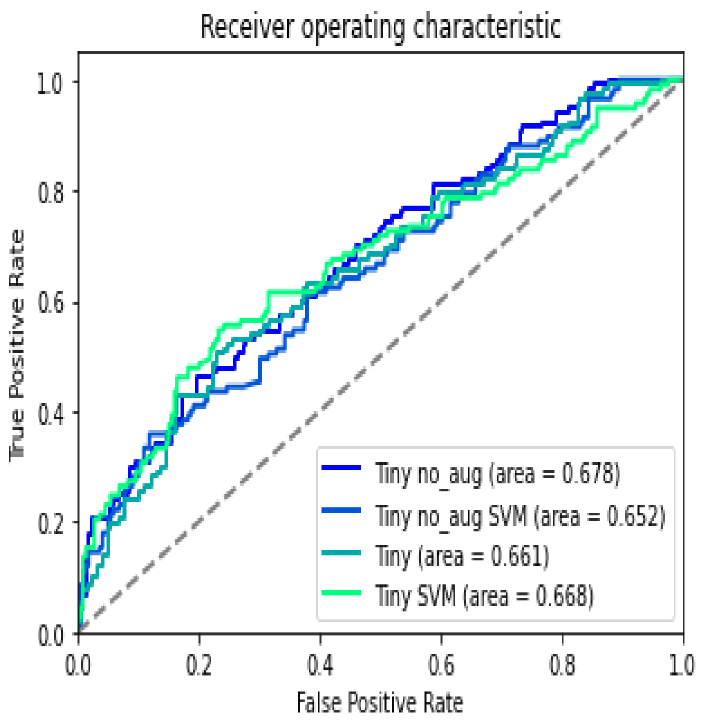


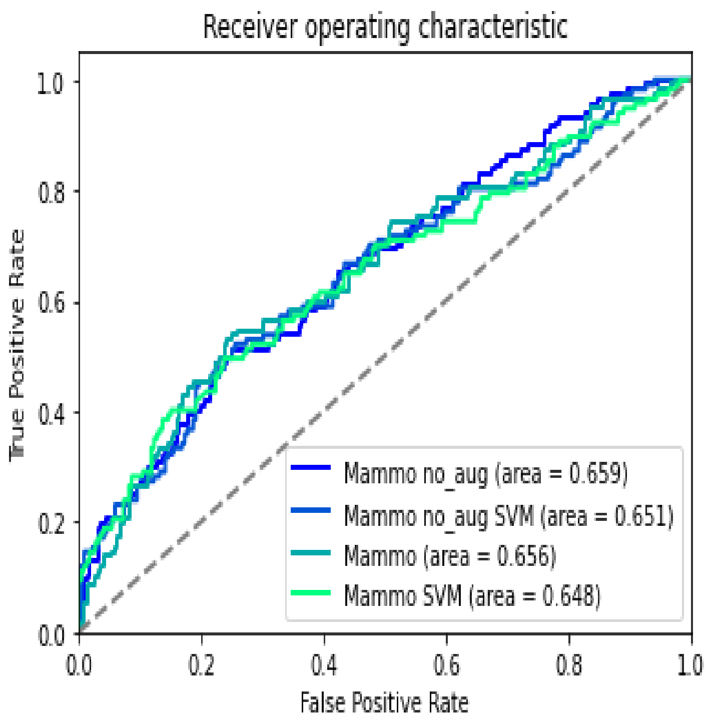
SVM Confusion Matrix:



Note that the *accuracy* in the *test set* is greater than that of *validation*. The motivation is that the classifier is clearly biased towards the classifications of *benign abnormalities*, and these are in higher proportion in the *test set* than in the *validation set*, producing higher *accuracy*. It should be noted that as in the case of the classification between *masses and calcifications*, the SVM allows to rebalance the classes if they are unbalanced in the original classifier.

RESULTS





It can be seen that all classifiers have similar performances and none is clearly better than the others, the only one that stands out slightly is *TinyNet without augmentation*, but the gap is not so significant and can be only a coincidence.

The results obtained are lower than those found in the literature, but it is not surprising given that in this project the data available were *fewer and of worse quality*. This will be confirmed by the fact that using the same *pre-trained* networks will result in worse results as well, even with the same training algorithms and parameters used.

TASK 3 – PRE-TRAINED CNNS

MODELS

As previously explained, in order to keep the number of experiments to be performed manageable, it was decided to focus the investigation on the three most used models that have achieved the best results in the literature, which will be briefly described below.

VGG-16 [10]

Is the network from which the inspiration for the creation of *MammoNet* was taken.

The authors of ***VGG-16*** investigated the effect of network depth while keeping the convolution filters very small. They showed that significant improvement can be achieved by pushing the depth to 16–19 layers. The input to the convolutional layer is a fixed-size 224 × 224 image. The image is passed through a stack of convolutional layers with *ReLU* activations where filters with very small receptive fields (3 × 3) were used. The convolution stride is also fixed to 1. Spatial pooling is carried out by five max-pooling layers, performed after some of the convolutional layers. Similarly to *AlexNet*, a stack of three *fully connected* layers is put on top of the convolutional part of the network. *The advantage of VGG is that, by stacking multiple convolutional layers with small-sized kernels, the effective receptive field of the network is increased, while reducing the number of parameters compared to using less convolutional layers with larger kernels for the same receptive field*.

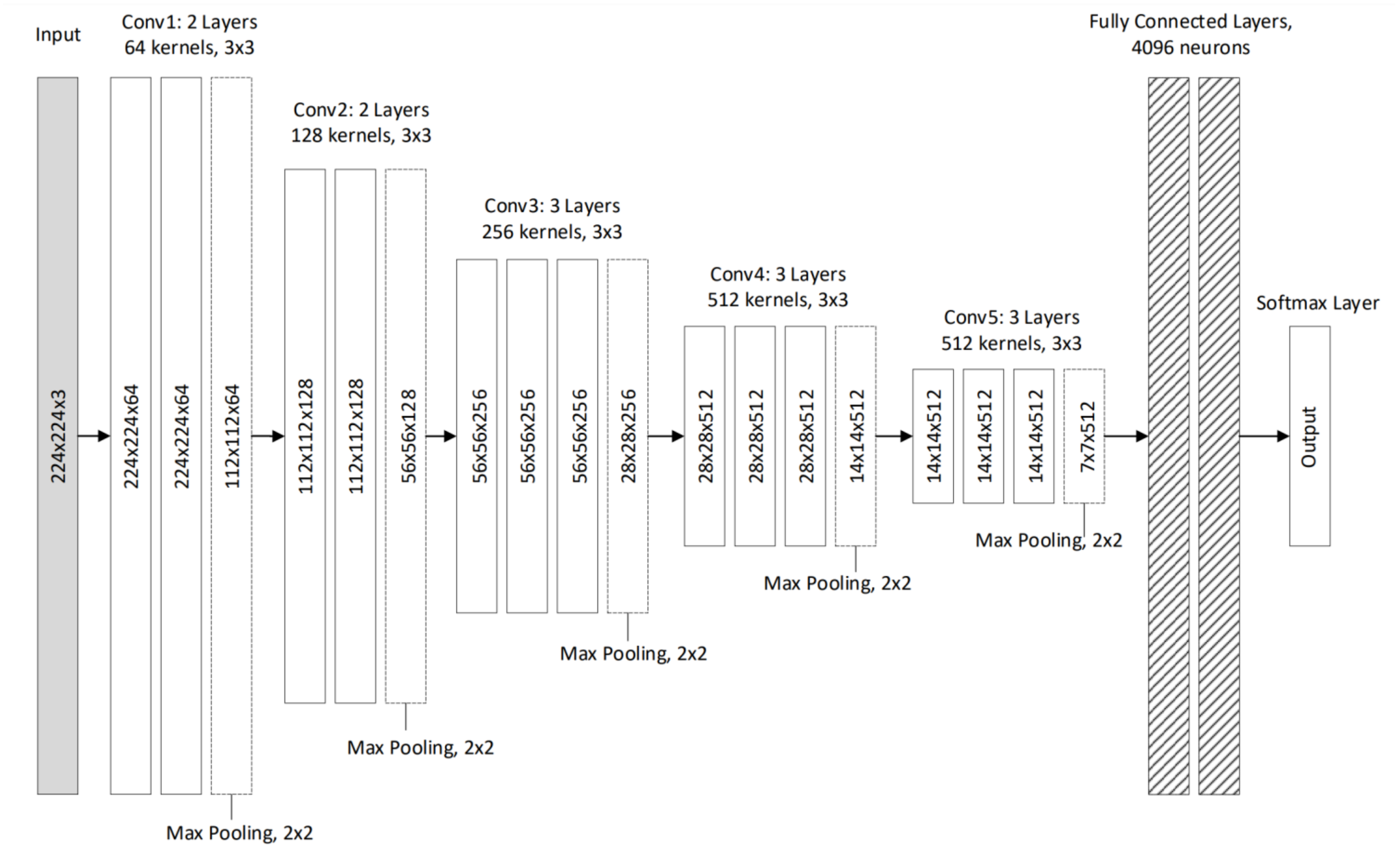


Figure 4: VGG16 architecture taken from [2]

INCEPTION-V3 [11]

GoogleNet is the first implementation using the ***Inception*** module (an example is shown alongside). Because of the huge variation in the location of the information, choosing the right kernel size for the convolution operation is tough for certain classification problems. A *larger* kernel is preferred for information that is distributed more *globally*, and a *smaller* kernel is preferred for information that is distributed more *locally*. The authors designed the *inception* module to have filters with multiple sizes to operate on the same level. The network essentially would get wider as well as deeper. In subsequent publications, a revised versions of the Inception module have been proposed, along with slightly modified network architectures. The authors proposed *Batch Normalization* (BN) and incorporated it into the Inception network.

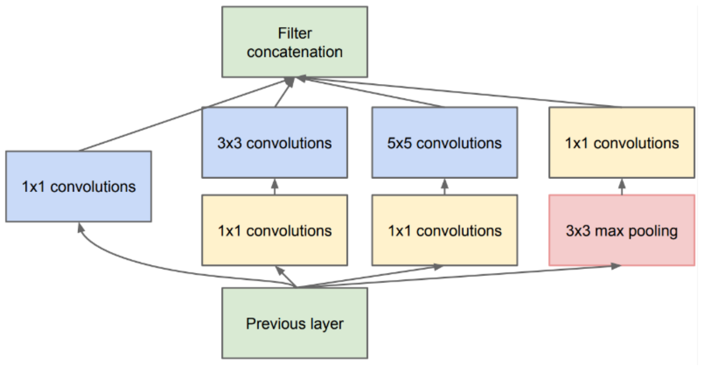
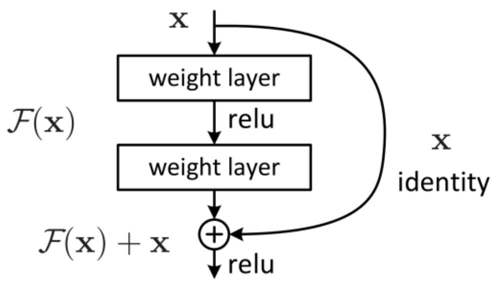


Figure 5: Exemple architecture of an Inception module from [11]

RESNET50 [12]

Residual networks (*ResNets*) consist of reformulated convolutional layers that are learning residual functions with reference to the inputs. The authors argue that this type of networks are easier to optimize and can be of significantly increased depth. The implementation of a *residual block* is straightforward: for every few convolutional layers a *shortcut connection* is added that runs parallel to these layers and implements the identity mapping. The output of the convolutional layers is then added to the output of the shortcut branch and the result is propagated to the subsequent block. Beside the use of shortcut connections, the network architecture is mainly inspired by the philosophy of *VGG* networks.


x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu
Figure 6: Residual learning building block from [12]

TRAINING AND RESULTS

INTRODUCTION TO THE EXPERIMENTS

The experiments were performed in the same way and with the same parameters as those used for *from-scratch* networks (Task 2). The only difference is that in this case the *grayscale* images were ***converted to RGB*** so that they could be used with *pretrained* networks on *ImageNet*, which was indeed composed of RGB images. This was done by simply tripling the *grayscale* channel for all three *RGB* channels, as is commonly done in the literature.

In this series of experiments, the comparison between the use of ***augmentation*** and not has not been repeated, as it is not possible to train such complex networks without them immediately *overfitting*, not using it.

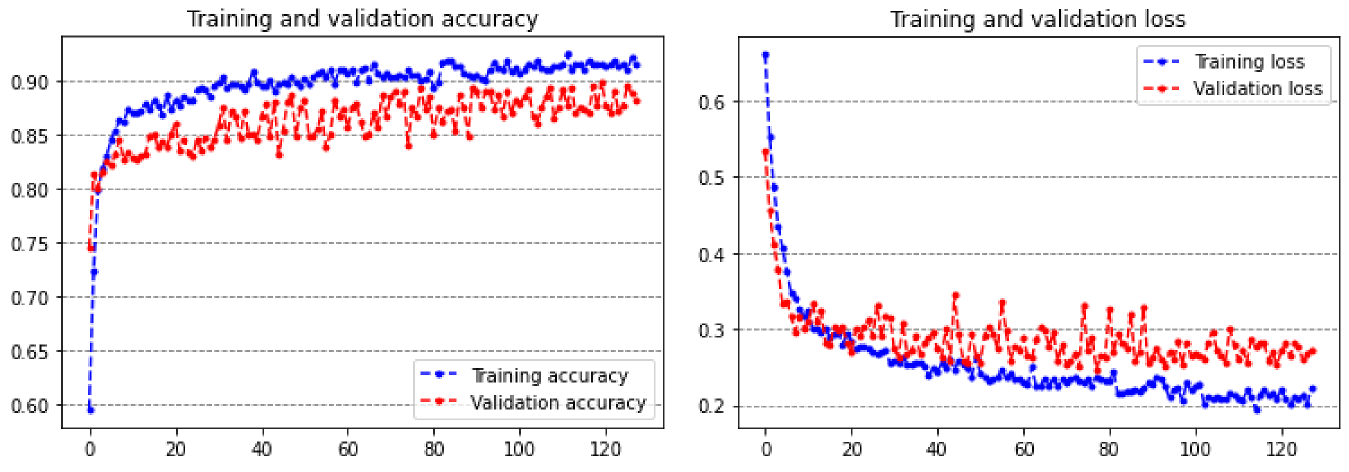
it has also been tested that the use of *SVM classifiers* on the extracted features does not lead to significant improvements, but at most a *rebalancing*, therefore to limit the number of experiments they will not be taken into consideration.

MASS-CALCIDICATION CLASSIFIERS

VGG16

Two *fully connected layers* with *4096* neurons attached to a sigmoidal neuron were used for the classification, the same combination used for *from scratch* networks.

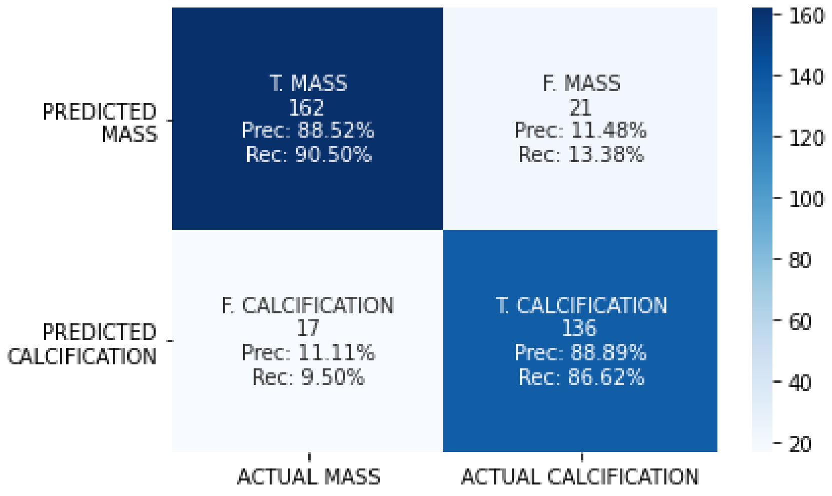
Transfer Learning



The training turns out to have a quite stable *logarithm curve*, if not for some anomalous peak in the loss. Is present a slight *overfitting,* but still contained.

Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
92.71%	0.184	89.37%	0.2943	**88.69%**	**0.2954**

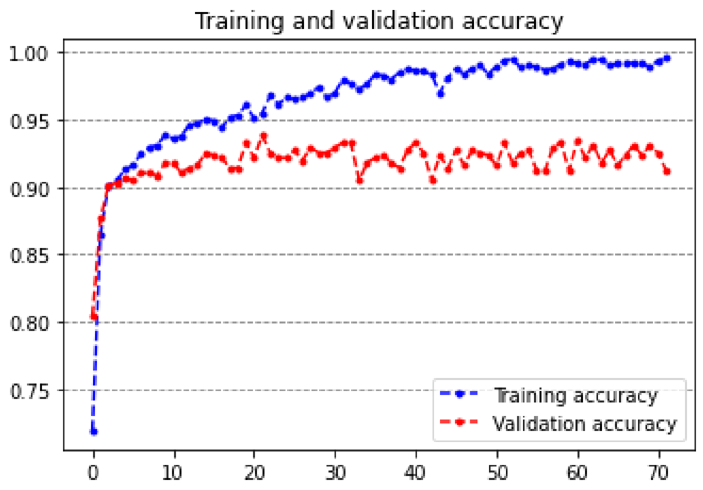
Confusion Matrix:

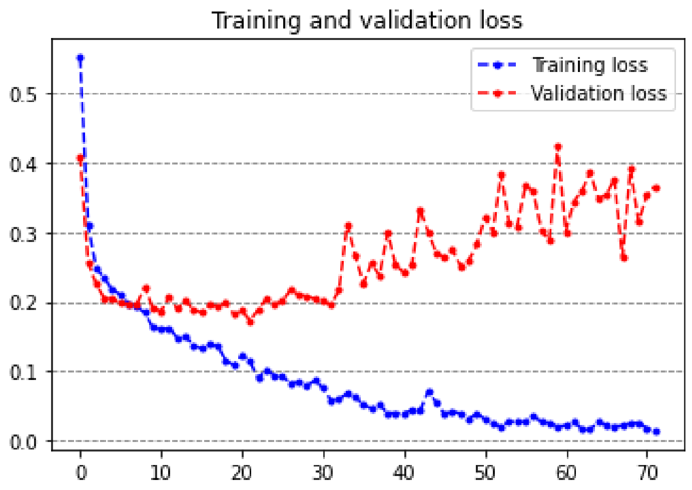


The classification appears to be slightly biased in favor of the *masses*, but still excellent. it should be noted that by performing only a transfer learning, and then using the features obtained from the training on *ImageNet*, the network is still suitable for the classification between *masses* and *calcifications*, although they are different from the *ImageNet* classes.

Fine Tuning

The various layers were "defrosted" one at a time to empirically test which was the best number for *fine tune* the network. The result is that the ***initial 8* layers** can be left frozen, as they have generic features useful for the problem, while the others can be trained starting from the *ImageNet* initialization of the *weights*.

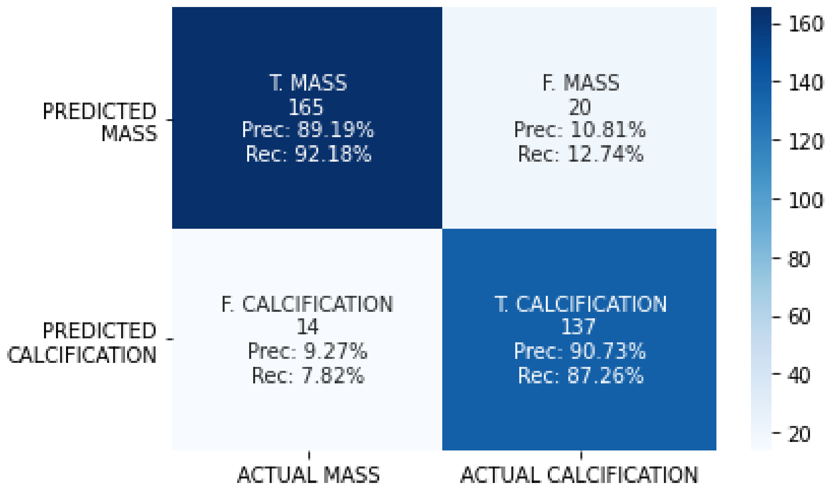




As generally happens for *fine tuning*, a few epochs are enough to achieve the optimum in training, after which *overfitting* quickly occurs.

Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
97.85%	0.0641	93.84%	0.2066	89.88%	0.2793

Confusion Matrix:

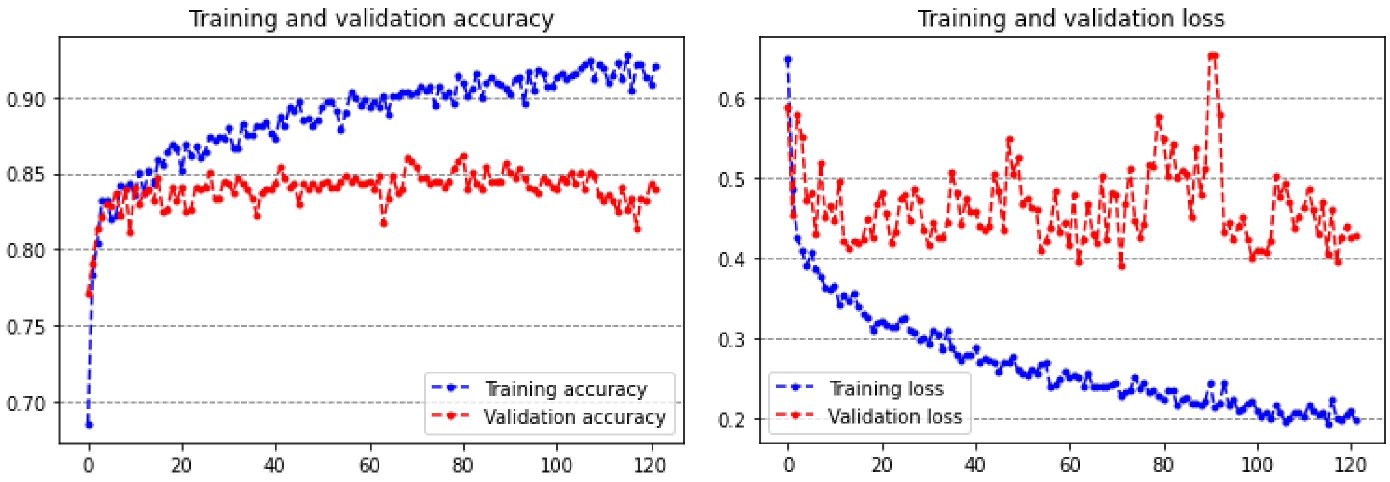


The classification seems excellent, albeit slightly biased in this case too in favor of the *masses*. it should be noted that the classification with the use of this network has similar performances to the *MammoNet* network, highlighting its excellent architecture for this type of problem.

INCEPTION-V3

Transfer Learning

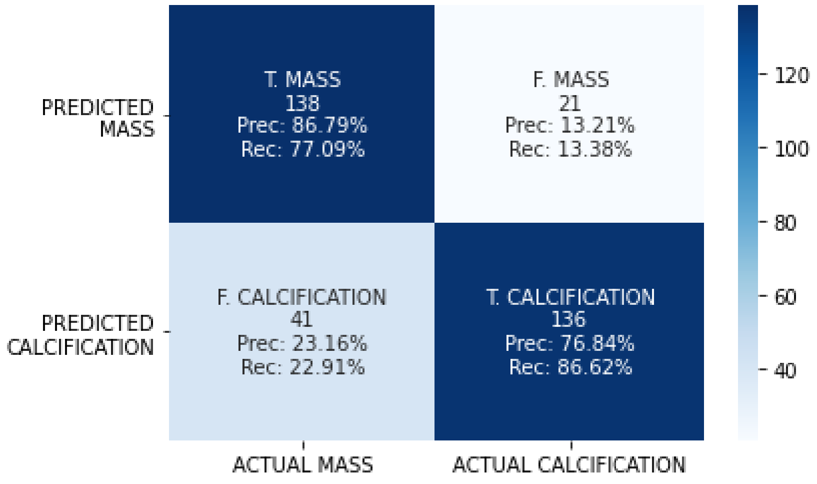
Two *fully connected layers* with *2048* neurons attached to a sigmoidal neuron were used for the classification. It was not possible to use a larger number of neurons as they would have led to memory problems, but the difference in the number of neurons has marginal influence.



The training is much more unstable than that of *VGG16*, moreover there is a strong *overfitting* from the beginning of the training. These factors lead the network to have one of the worst results so far obtained in this type of classification.

Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
93.27%	0.1836	84.7%	0.3719	81.55%	0.4152

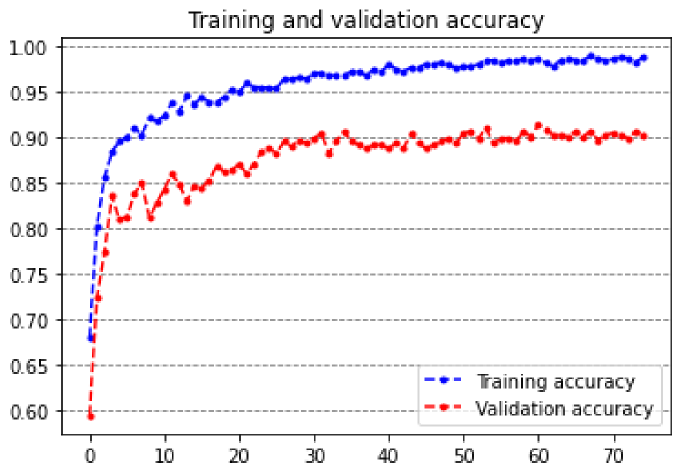
Confusion Matrix:

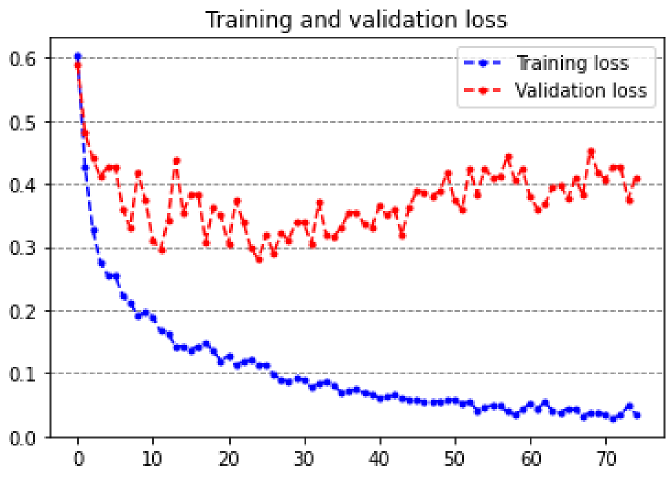


The network appears to be biased towards the classification of *calcifications*.

Fine Tuning

Empirically evaluating the various possibilities, the one with the best results seems to be the complete "defrosting" of the whole network, despite its enormous size. therefore, a ***weights initialization*** technique using ImageNet database was used.

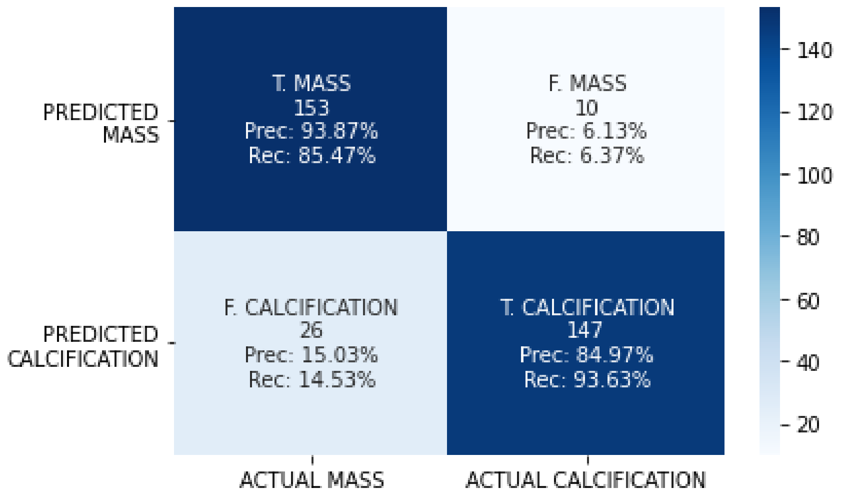




Even if there is a strong *overfitting* from the beginning of the training this has significant effects only after thirty epochs allowing to reach optimal loss values in the meantime.

Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
95.47%	0.117	88.81%	0.3355	89.29%	0.2834

Confusion Matrix:



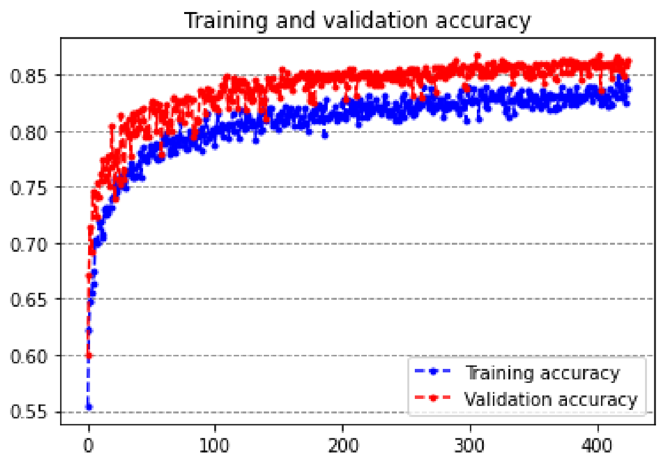
Although *Inception-V3*'s *transfer learning* results were disappointing, *fine tuning* proved to be very productive in this case. The classification was balanced, and the network went from being poor to be very good in terms of *accuracy*. However, a slight *imbalance* in favor of *calcifications* remains present.

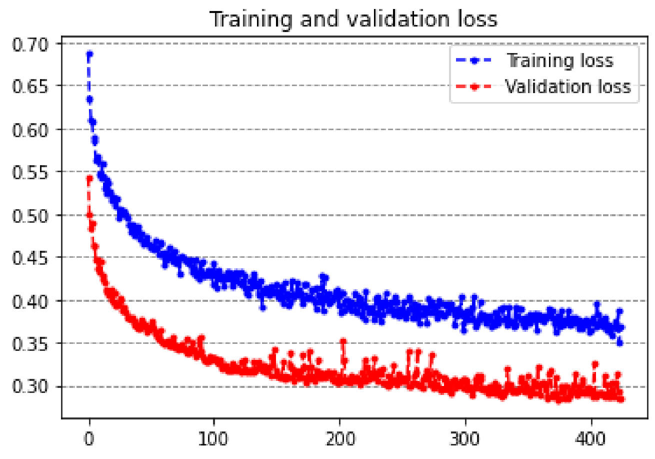
RESNET50

Given the exorbitant number of neurons in the last convolutional layer of the network, it was necessary to use only *512* neurons in the fully connected layers. This makes the network usable at the cost of a negligible loss in accuracy.

It should be noted that there will not be present a *fine-tuning* section for this network, as all the trainings were unsuccessful due to the nature of the network. This is also highlighted in the literature [1] and is not an anomalous result.

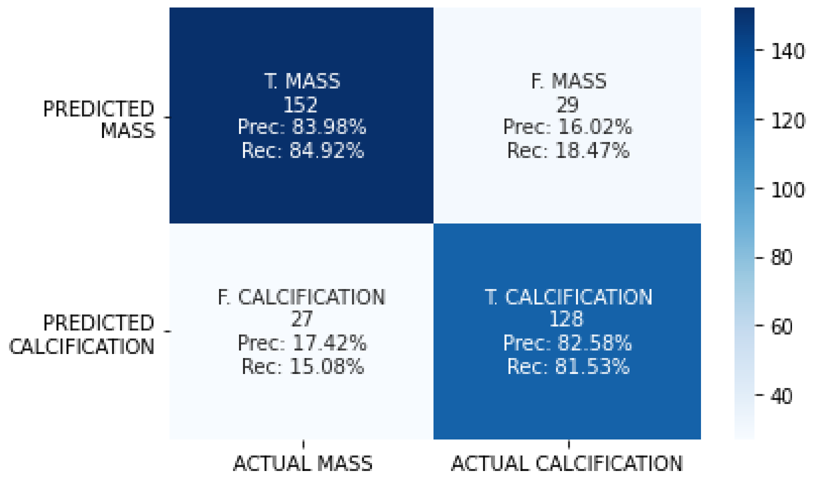
Transfer Learning





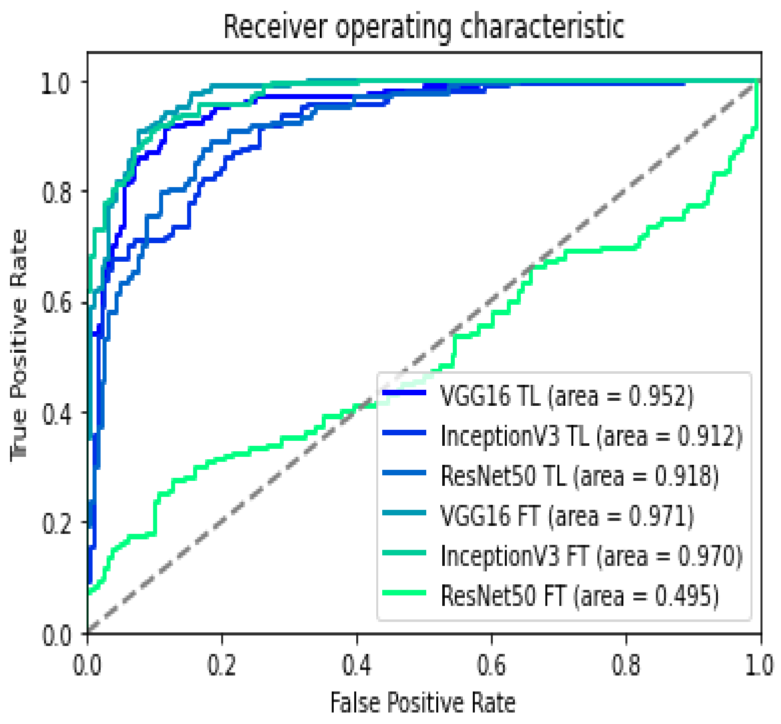
The training turns out to have a very stable but at the same time anomalous trend. The *validation* has better values than *training*, an explanation may be that of the use of *augmentation* which had similar effects in networks *from scratch*. It is also possible to notice a strong *overfitting* from the beginning which probably prevented a good configuration from being achieved.

Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
87.34%	0.2966	86.38%	0.3388	**83.33%**	**0.3704**



Confusion Matrix:
The network has mediocre performance if compared with the other networks used, even *from scratch*, but it could be useful in an *ensemble* method given its diversity.

RESULTS



The performance of the *ROC curves* is optimal for all classifiers (except *InceptionV3 FT*), but it is evident how ***VGG16*** and ***InceptionV3*** in ***fine tuning*** are structurally better. Their curves, in addition to having a very high *AOC*, are also very balanced, making it possible to move the classification threshold as preferred if a greater recall is required for one of the two classes.

Given that in the *benign-malignant* classification the *VGG16* network was also found to be the best and it will be used for ***task 4*** of this project.

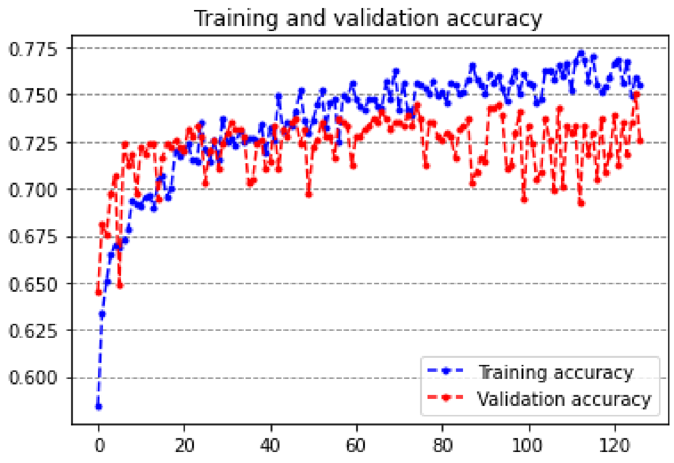
It should be noted that as previously described and as also highlighted in the literature [1], is not possible to train *Inception-V3* in *fine tuning*, leading to a random classifier.

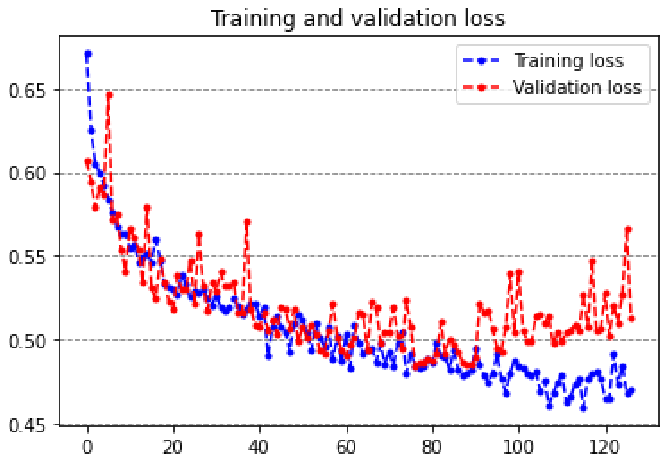
BENIGN-MALIGNANT CLASSIFIERS

VGG16

Two *fully connected layers* with *4096* neurons attached to a sigmoidal neuron were used for the classification. The same combination used for *from scratch* networks

Transfer Learning

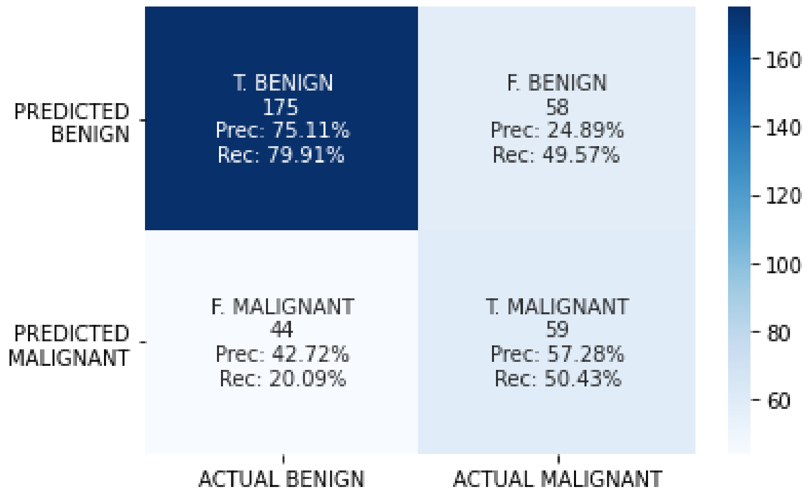




The progress of the training is in line with that of *MammoNet*, however the overfitting is less accentuated allowing to reach higher levels of accuracy.

Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
77.99%	0.4412	71.27%	0.5225	**69.64%**	**0.5891**

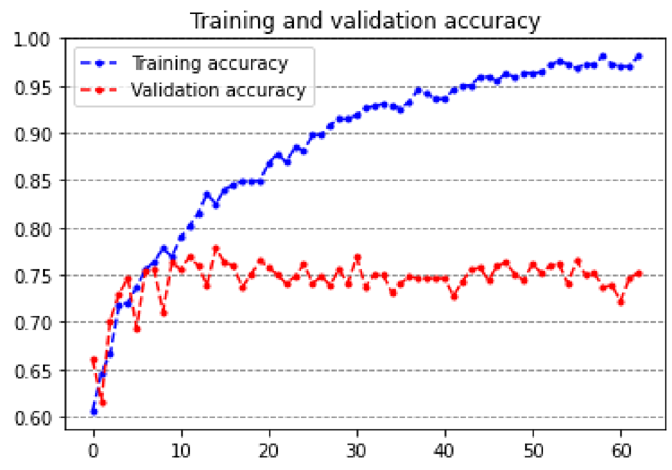
Confusion Matrix:

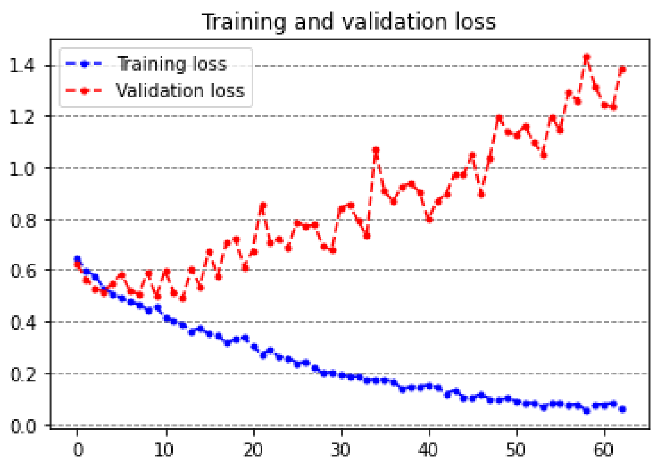


As was the case for the networks *from scratch*, the classifier appears to have a greater confidence in the classification of *benign abnormalities* rather than *malignant abnormalities*. However, there is an improvement in the *precision* of the classification of *malignant abnormalities*, even if the *recall* remains around 50%.

Fine Tuning

The various layers were "defrosted" one at a time to empirically test which was the best number for *fine tune* the network. The result is that the ***initial 8* layers** can be left frozen, as they have generic features useful for the problem, while the others can be trained starting from the *ImageNet* initialization of the *weights*.

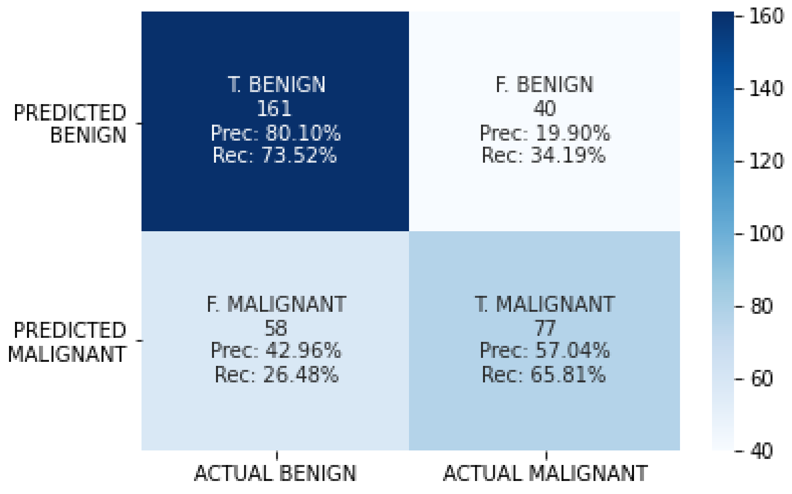




As is well known, fine tuning quickly leads to *overfitting*, therefore the optimal combination of the values of the weights is reached in the first epochs.

Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
85.28%	0.3252	75.93%	0.5381	**70.83%**	**0.6135**

Confusion Matrix:

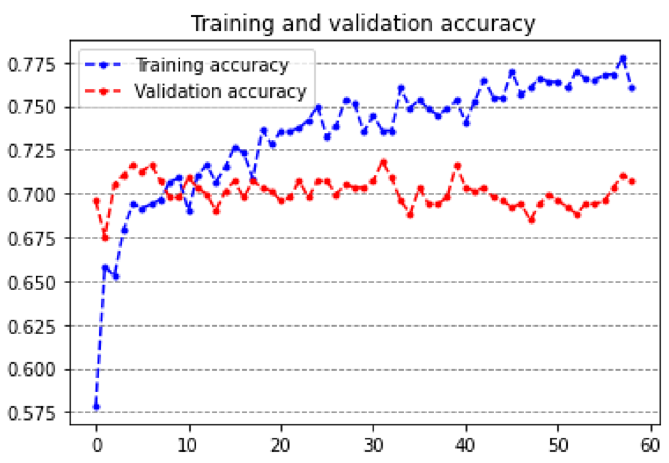


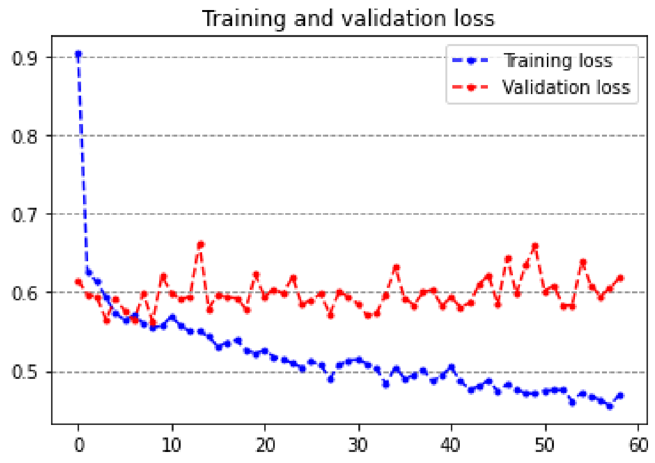
it is possible to see how *fine tuning* actually has a fair impact on the classification. The classification is in fact better than that with *transfer learning* alone. This is not evidenced by greater *accuracy* due to the imbalance of the test set towards *benign abnormalities*, while the real improvement is in the classification of *malignant abnormalities*, finally no longer random.

INCEPTION-V3

Two *fully connected layers* with *2048* neurons attached to a sigmoidal neuron were used for the classification. It was not possible to use a larger number of neurons as they would have led to memory problems, but the difference in the number of neurons has marginal influence.

Transfer Learning

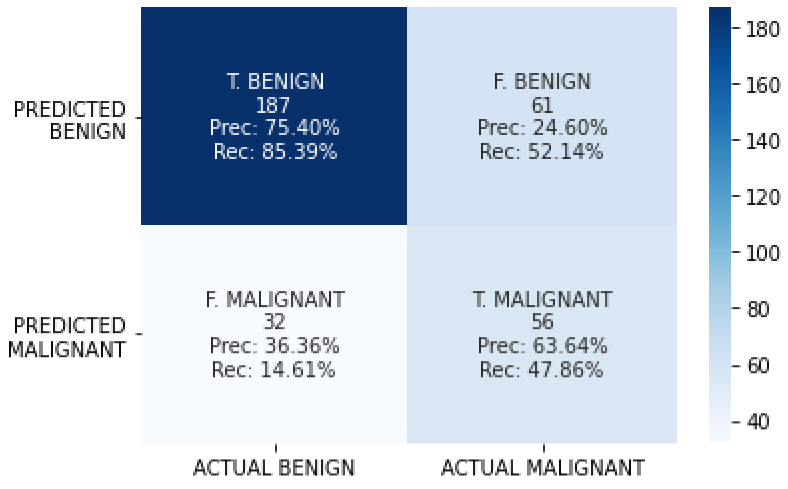




it is possible to notice how the training reaches the optimum within the first 10 epochs even if only classification layers are trained while the others remain frozen, then it quickly goes in *overfitting*. This effect is probably due to the great complexity of the network.

Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
71.96%	0.5279	69.78%	0.5675	**72.32%**	**0.5921**

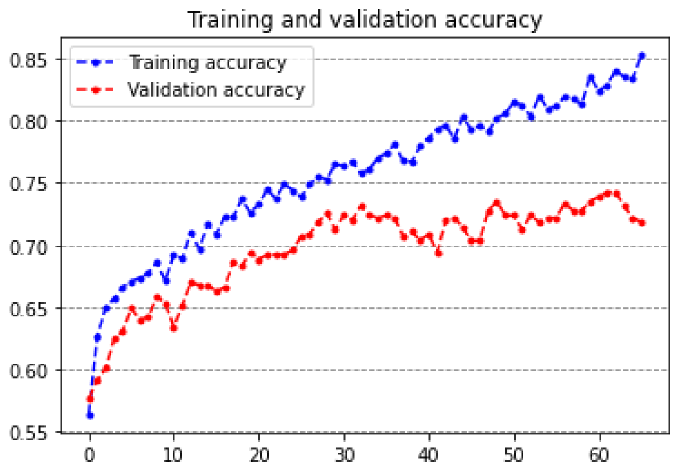
Confusion Matrix:

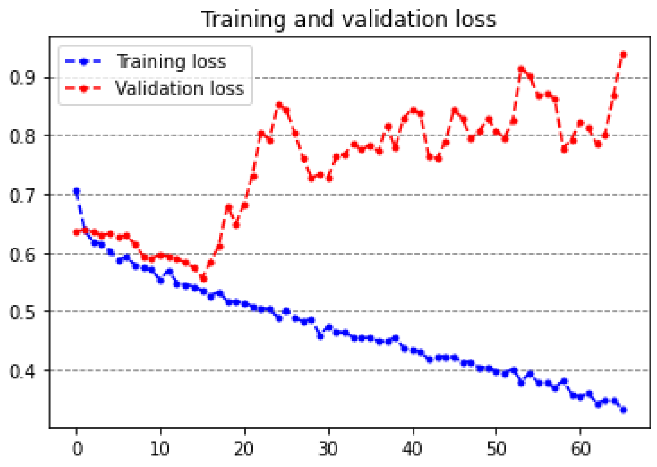


It can be noticed that the network is biased towards the classification of *benign abnormalities,* this leads to greater *accuracy* than the previous network, but as already mentioned it is an effect due to the imbalance of the classes in the test set. This is also highlighted by a lower level of accuracy in *validation*, since the set is more balanced.

Fine Tuning

Empirically evaluating the various possibilities, the one with the best results seems to be the complete "defrosting" of the whole network, despite its enormous size. therefore, a ***weights initialization*** technique using ImageNet database was used.

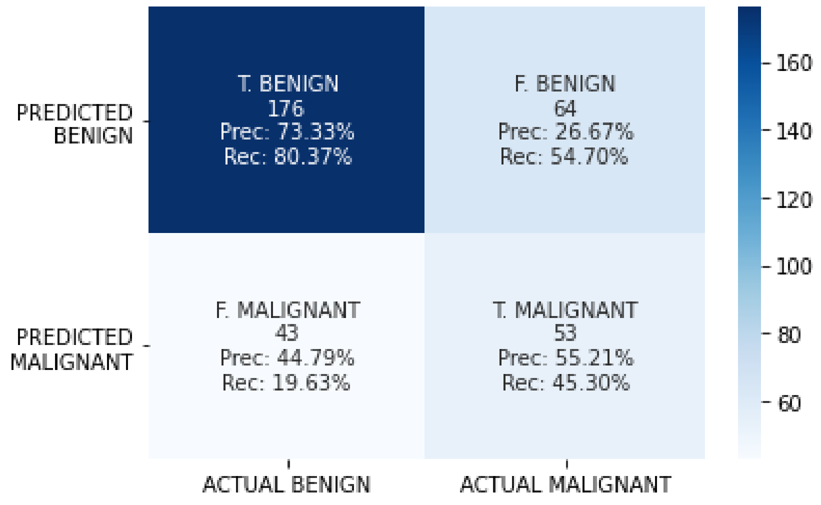




As often happens for fine tuning, the training improves for the first few epochs and then quickly overfitting, because of the high number of weights.

Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
71.17%	0.5368	66.23%	0.6031	**68.15%**	**0.588**

Confusion Matrix:



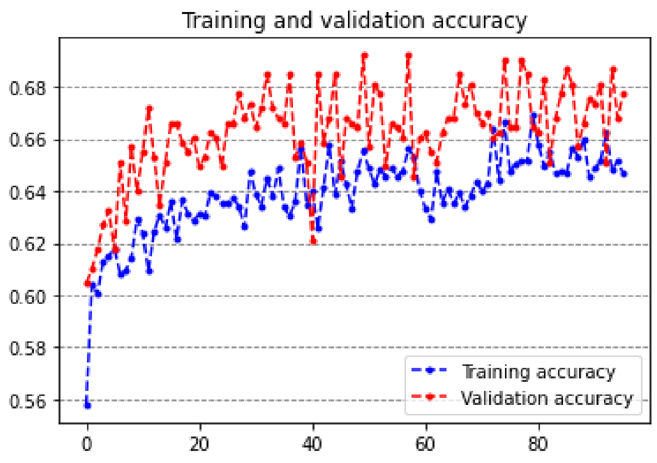
The *fine tuning* in this case does not seem to have been successful since the result is a worse classifier than using *transfer learning*. However, looking at the results of the ROC curve at the end of the section, is possible to see how the classifier is in reality structurally slightly better.

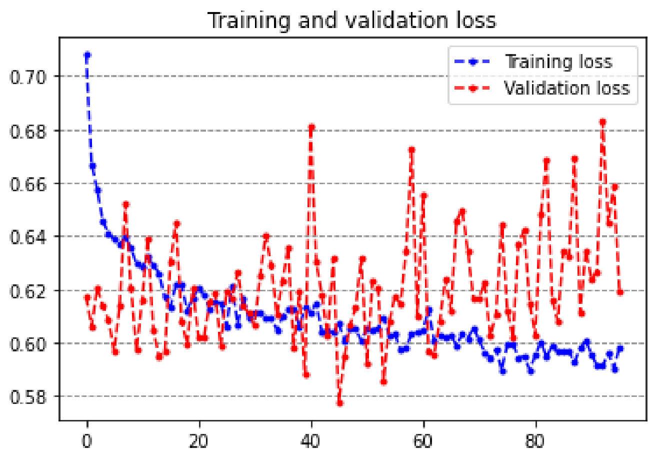
RESNET50

Given the exorbitant number of neurons in the last convolutional layer of the network, it was necessary to use only *512* neurons in the fully connected layers. This makes the network usable at the cost of a negligible loss in accuracy.

It should be noted that there will not be present a *fine-tuning* section for this network, as all the trainings were unsuccessful due to the nature of the network. This is also highlighted in the literature [1] and is not an anomalous result.

Transfer Learning

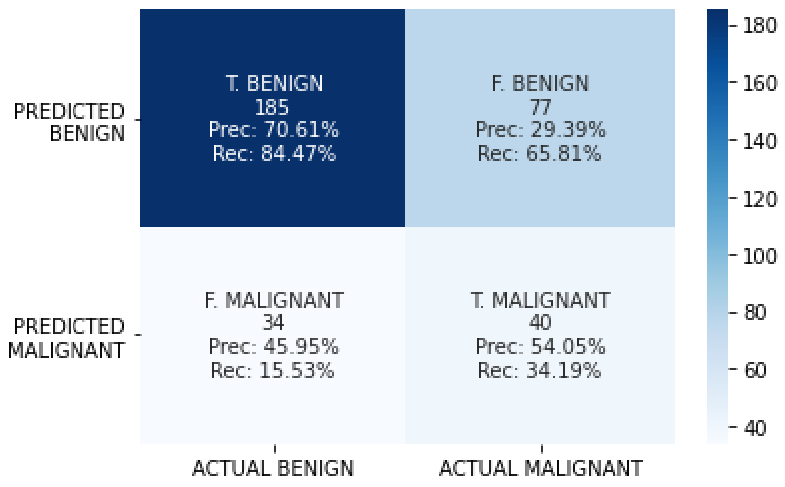




it is possible to notice how the network is very *unstable* in the training phase, despite the use of a very low learning rate. Nevertheless, the results still seem to be in line with those of other pre-trained networks.

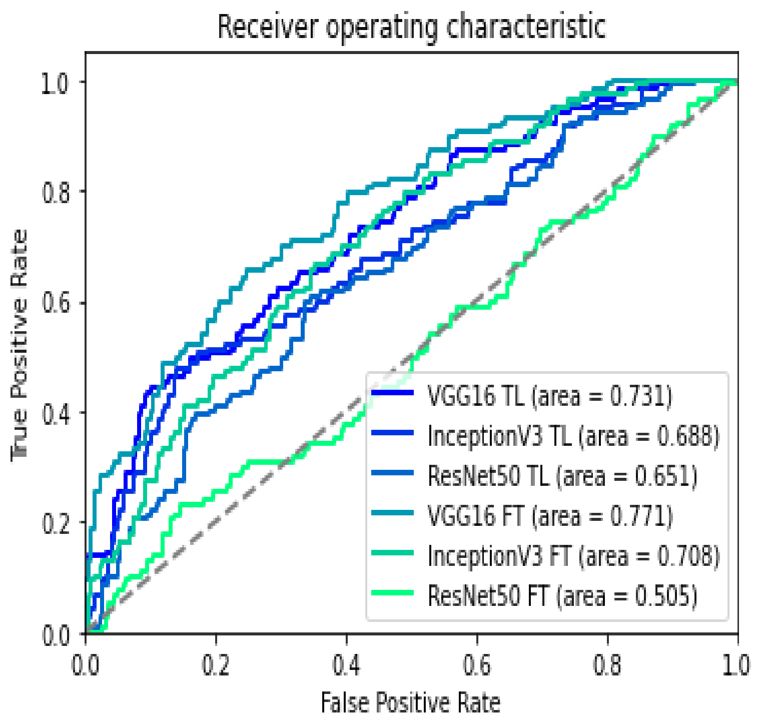
Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
75.33%	0.4647	70.9%	0.5567	**72.32%**	**0.5744**

Confusion Matrix:



The good accuracy of this network, the best among those in *transfer learning*, is, as often happened, due to a strong *imbalance* of the classifier towards the majority class of *benign abnormalities*. The classifier turns out to be in reality among the worst, as evidenced by the *ROC* curve shown below.

RESULTS



The direct comparison using the *ROC* curves highlights how ***fine tuning*** is the winning choice for this type of classification, even in comparison to *from scratch* classifiers. However, the results are still considerably below the average reported in the papers, even if major improvements have been made. This difference, as already highlighted, is probably due to the *dataset* used, as it would not be explained otherwise since a lot of the combinations reported in the literature have been tested.

Among all the classifiers, however, ***VGG16*** in fine tuning stands out, this network will therefore be used as a reference to test the use of the *baselines* in ***Task 4***.

Is it possible to notice how the *fine tuning* of *ResNet50* has no effect and creates a network that classifies randomly. This is also highlighted in the literature [1] and is probably due to the particular structure of the network itself, as well as its considerable depth.

TASK 4 – BASELINE CNNS

MODELS

CLASSICAL SIAMESE CNNS

A ***Siamese Neural Network*** is a class of neural network architectures that contain two or more *identical* subnetworks. They have the same configuration with the same parameters and weights. Parameter updating is mirrored across both sub-networks. It is usually used to find the *similarity* of the inputs by comparing its *feature vectors*.

Siamese networks are nice for ***ensemble***: Given that their learning mechanism are somewhat different from base classifiers, ensemble them can lead to better results than ensemble multiple correlated models.

To exploit the ***baseline patches*** it was necessary to alter the typical structure of these networks in a similar way done in [13]. Instead of calculating the *distance* between the *feature vectors* as in classical Siamese, in this case the features are *combined* in order to take into account at the same time the *abnormalities* and the *baselines* with the hope of obtaining additional information from the baselines themselves.

SIAMESE DIFFERENCE

In this case, as shown in the figure, instead of calculating the *similarity* of the *feature vectors* of the networks, the ***difference in absolute valu****e* is performed, in this way a *"net" feature vector* is obtained that hopefully contains only the characteristic features of the *abnormality*.

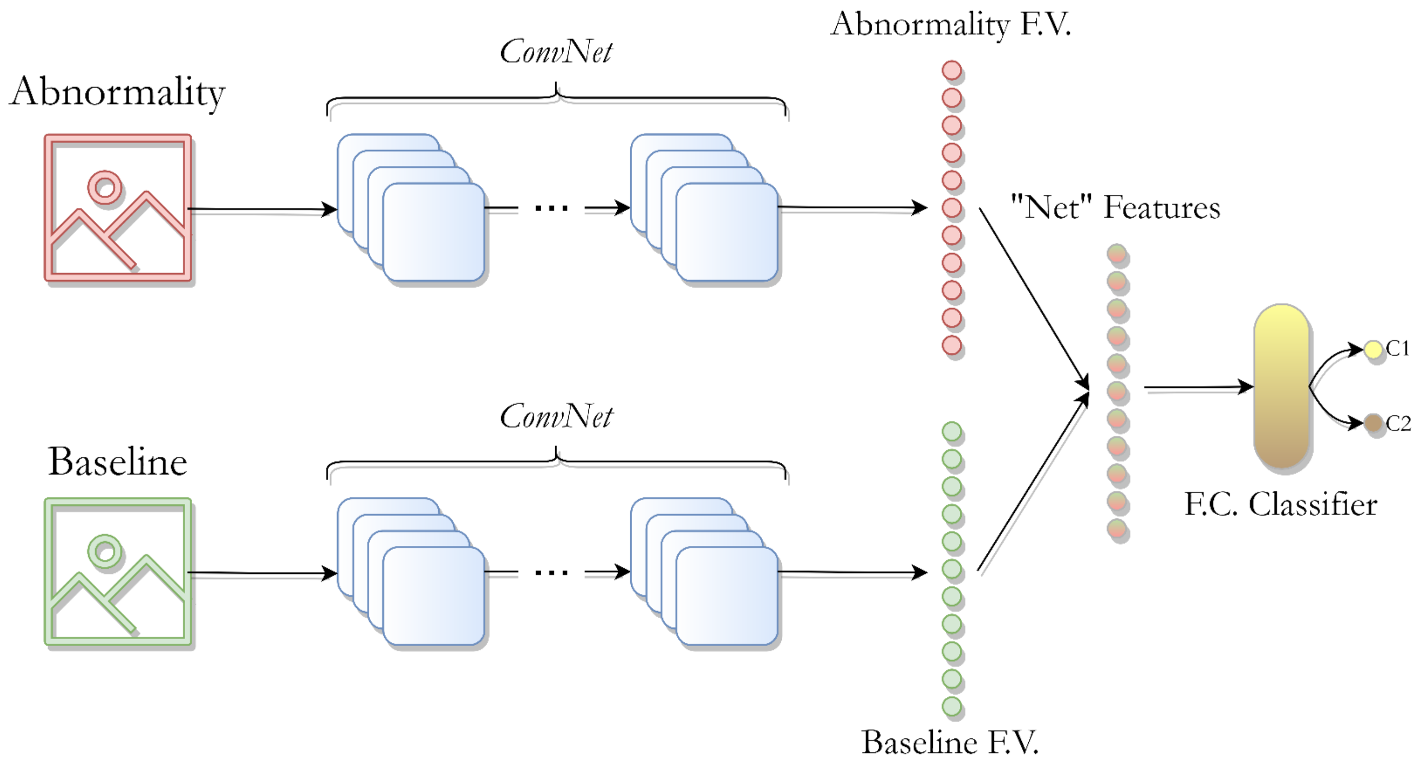


Figure 7: Siamese features vectors difference schema

SIAMESE CONCATENATION

In this case instead, the *features vectors* of the *abnormality* and the *baseline* are ***concatenated***, so that the network can decide independently which is the best way to combine them.

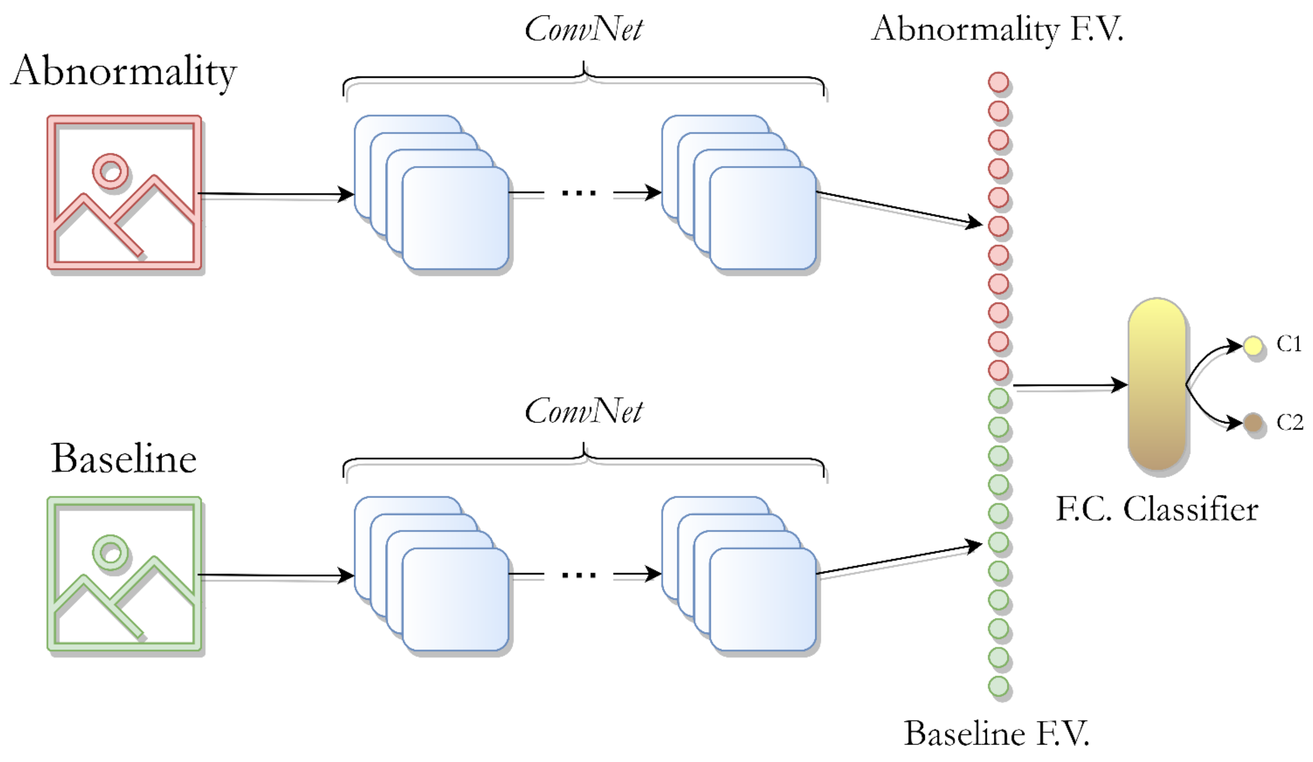


Figure 8: Siamese features vectors concatenated schema

TRAINING AND RESULTS

INTRODUCTION TO THE EXPERIMENTS

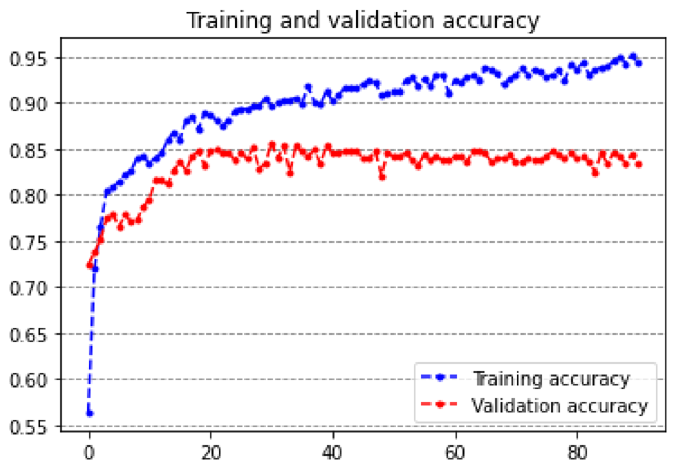
To train the Siamese networks, it was taken as *ConvNet* ***VGG16 in fine tuning***, since it turned out to be the best network both for the classification of *benign and malignant* abnormalities and for the classification between *masses and calcifications*.

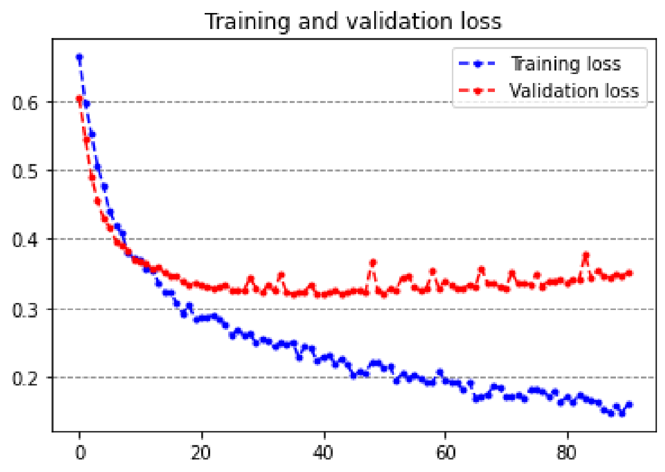
To the *features vector* elaborated according to the scheme used, among those described above, *two fully connected layers* of *4096* neurons each and a *sigmoidal classification layer* composed of a neuron were connected. The scheme is the same as that used for the other *ConvNet* and includes the use of *dropouts* too.

The experiments were performed in the same way and with the same parameters as those used for *pre-trained* networks (***Task 3***).

MASS-CLACIFICATION CLASSIFIERS

SIAMESE DIFFERENCE

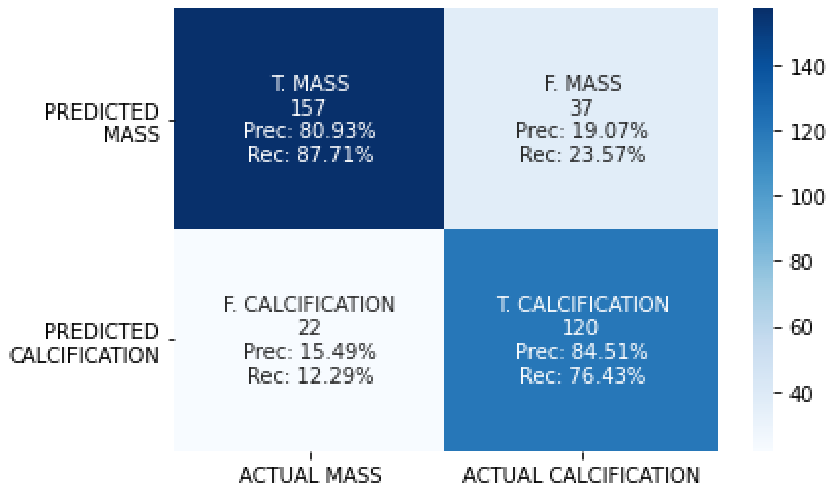




The use of a *Siamese network* seems to regulate the progress of the training. As happens when performing *fine tuning*, *overfitting* soon occurs, despite the use of *augmentation* and *dropout*, due to the large number of *weights* in the network.

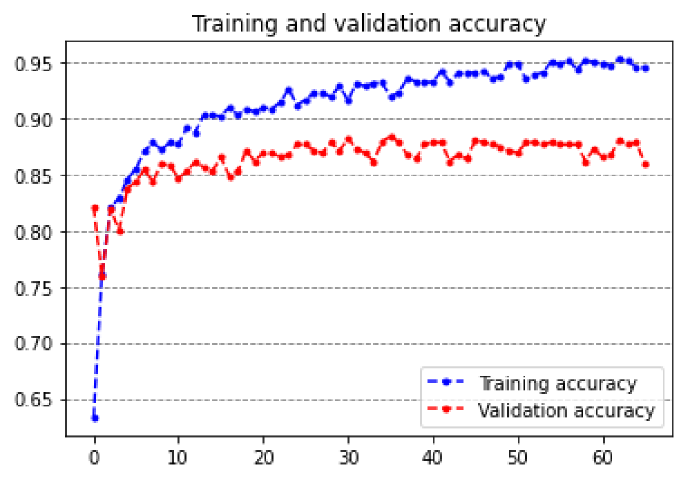
Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
91.87%	0.2035	84.51%	0.3712	**82.44%**	**0.4357**

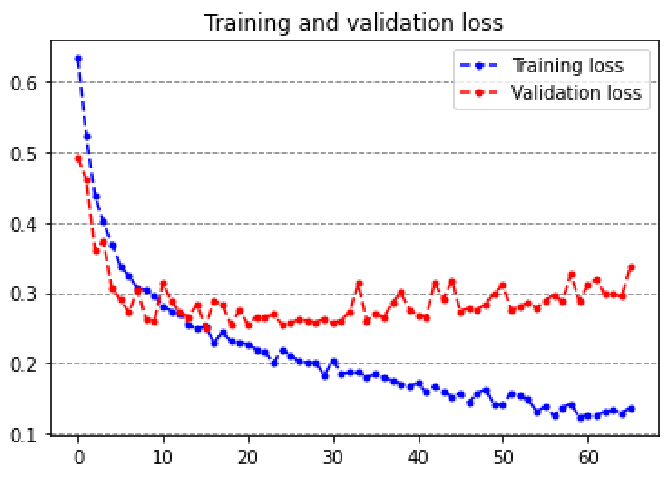
Confusion Matrix:



Subtracting the *features vector* of the *baseline* from that of the *abnormality* did not lead to notable results even if it is actually working. The classification is biased towards the *masses* and the total *accuracy* is not better than that of *TinyNet*, the smallest of *from scratch*'s networks.

SIAMESE CONCATENATION

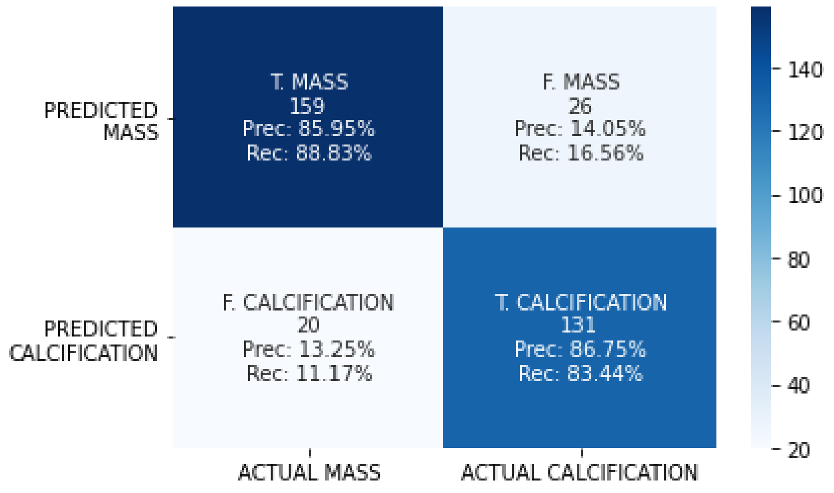




The training trend is similar to that previously described for the *features vectors' difference Siamese network*

Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
91.59%	0.2078	86.57%	0.2918	**86.01%**	**0.3053**

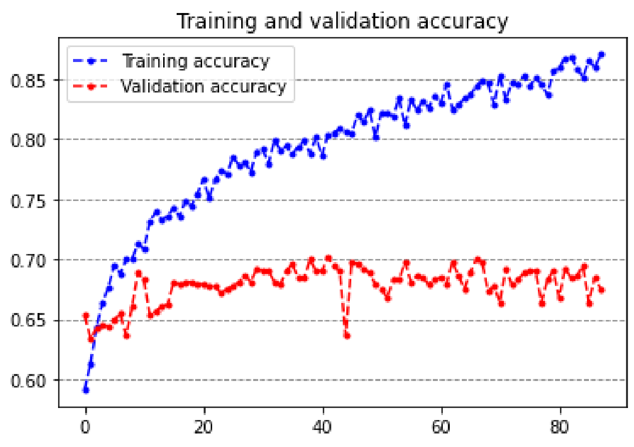
Confusion Matrix:

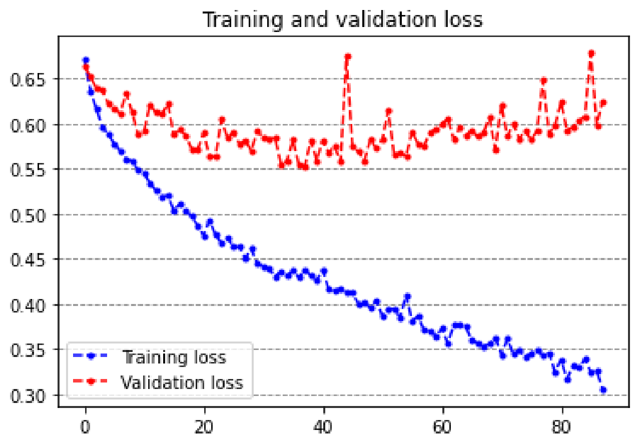


This version, however, appears to have better performance, even if the classification remains slightly biased towards the masses.

BENIGN-MALIGNANT CLASSIFIERS

SIAMESE DIFFERENCE

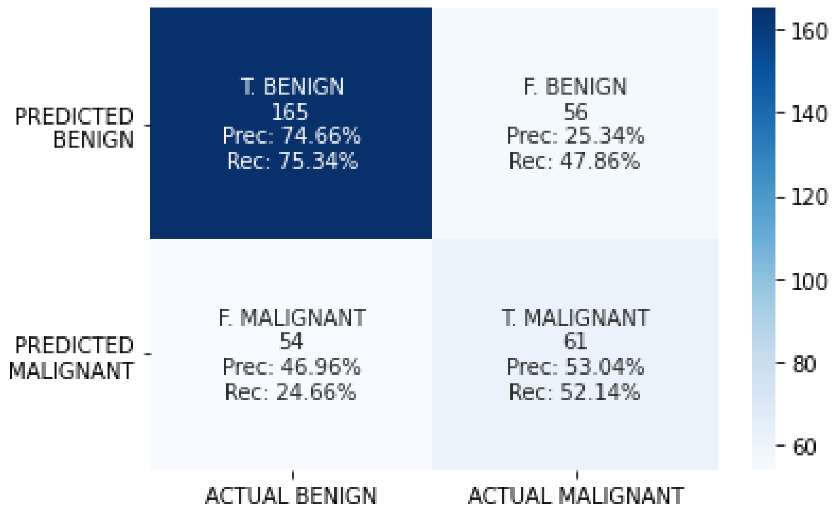




In the *malignant-benign* case, *overfitting* occurs from the first training's epochs not allowing to reach optimal levels of *accuracy*, even with the use of *dropout* and *augmentation*.

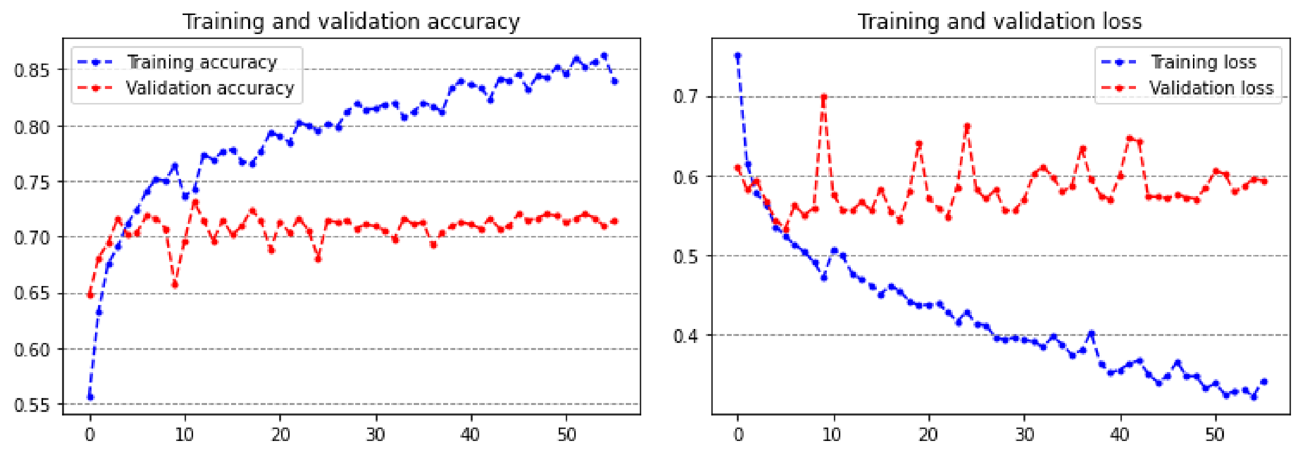
Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
84.21%	0.3836	68.47%	0.5852	**67.26%**	**0.5987**

Confusion Matrix:



The classification is biased towards the classification of *benign abnormalities,* as is now known to occur in this type of classification. The use of *baselines* does not alter the way in which classes are interpreted by the network.

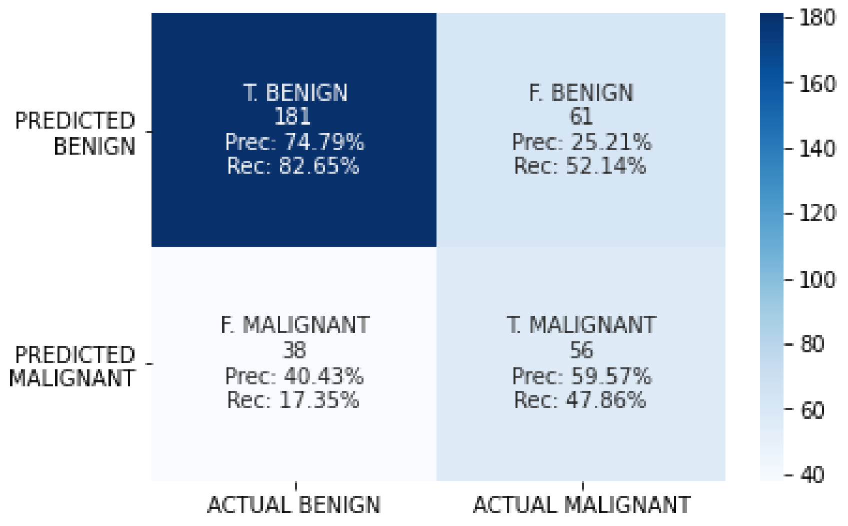
SIAMESE CONCATENATION



Also in this case the network is trained only in the early epochs, however it must be considered that it is a *VGG16 fine tuning* scheme that even in the non-Siamese version had this tendency.

Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
76.12%	0.4827	70.34%	0.5531	**70.54%**	**0.5582**

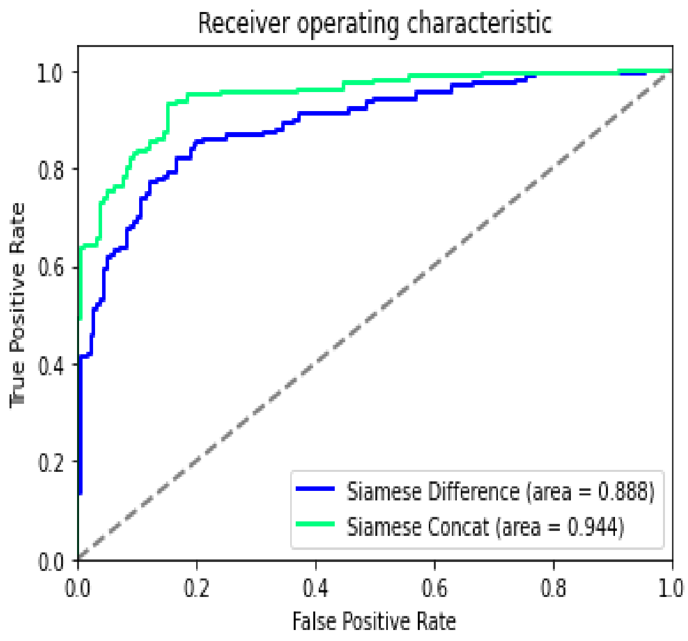
Confusion Matrix:



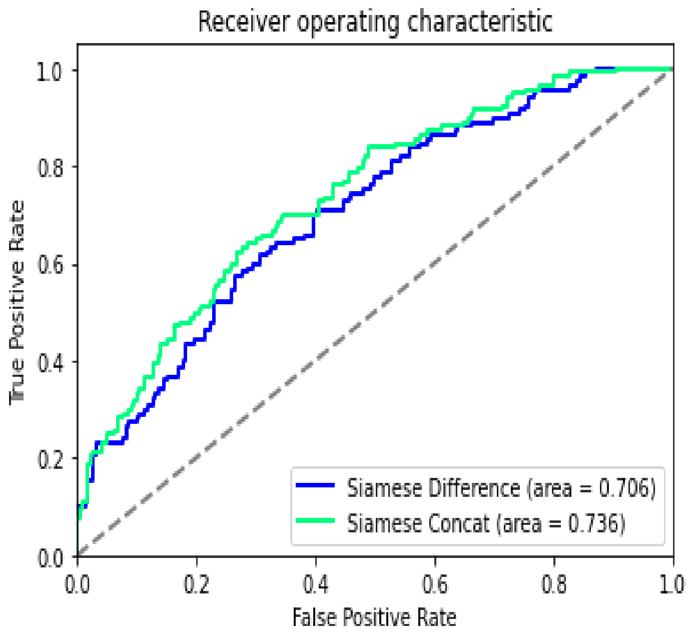
Compared to the version that makes the difference between the *features vectors*, this has a better *accuracy* even if the imbalance in favor of the prediction of *benign abnormalities* is the cause. This confirms that the use of *baselines* in a scheme with a *Siamese network* does not help in the classification.

RESULTS

Mass-Calcification Classification:



Benign-Malignant Classification:



Is not surprising that the use *Siamese networks* and *baselines* does not increase performance. In fact, this approach is not even attempted in the literature, most likely because instead of using *baselines* it would be enough to use a *larger patch* or multiple images of the same abnormality using *different views* as done in [13]. It can also be noted that it is better to let the network decide how to combine the *features vectors* by passing the *concatenated* vectors and not their *difference* to the *fully connected layers*.

The use of this type of networks, however, can be useful to improve the classification if used in an ***ensemble*** with the network of which they are composed. In fact, an *ensemble* with the Siamese version would probably be more successful, as the classifications would be less correlated than simply using different *weights' initializations*.

Following the results obtained, it was also tested the possibility that the use of two ensemble (with *different weights*) networks, to combine the *baselines* with the *abnormalities,* could obtain better results. However, the experimental results were found to have similar performances as shown above, so the use of *baselines* probably does not add useful information to the classification.

TASK 5 – ENSEMBLE CNNS

MODELS

There is a great variety of methods to perform the ***ensemble*** of neural networks, in order to improve the performance of the networks taken individually.

The use of *ensemble* is effective if the networks are ***unrelated***. Therefore, it is generally preferred to use completely different networks instead of the same network initialized for example with different *weights*. This is explained by the fact that related networks tend to classify in the same way and therefore classification has a marginal benefit in putting their views together. If, on the other hand, the networks are of very different typologies, they have a basically different vision of the problem, therefore the classification benefits from combining the strengths of the various views of the networks in *ensemble*.

Given the reasons listed above, it was decided to use the best version of the *pretrained* CNNs between the *fine tuning* or *transfer learning* ones together with the *Siamese networks*, since these networks should have very different visions of the problem. The chosen networks are therefore ***VGG16 FT, InceptionV3 FT, ResNet50 TL, and the Siamese ones.***

WEIGHTED ENSEMBLE

The first type of *ensemble* tested is an ensemble ***weighted*** for the ***accuracy*** of the various networks (as similarly used in [3]), in order to give priority to the networks that generally perform better, but still give the possibility to the others to influence classification if the *confidence* with which they classify a certain class is very high respect to the other. For the same reason this technique has been preferred to the *majority voting*, as a vote does not consider the ***confidence*** by which a network votes for a certain class.

STACKED ENSEMBLE

However, finding the right combination of *weights* and *networks* can be a problem on its own, and depending on how those factors are combined it can even worsen the classification.

Why not let another network decide what is the best way to combine CNNs?

This approach is called ***stacked generalization*** and consists of training an entirely new model to learn how to best combine the contributions from each *sub-model*. By letting a neural network decide is possible to be sure to improve the classification or in the worst-case scenario to keep that of the best classifier, without worrying about which networks to use in the ensemble and how.

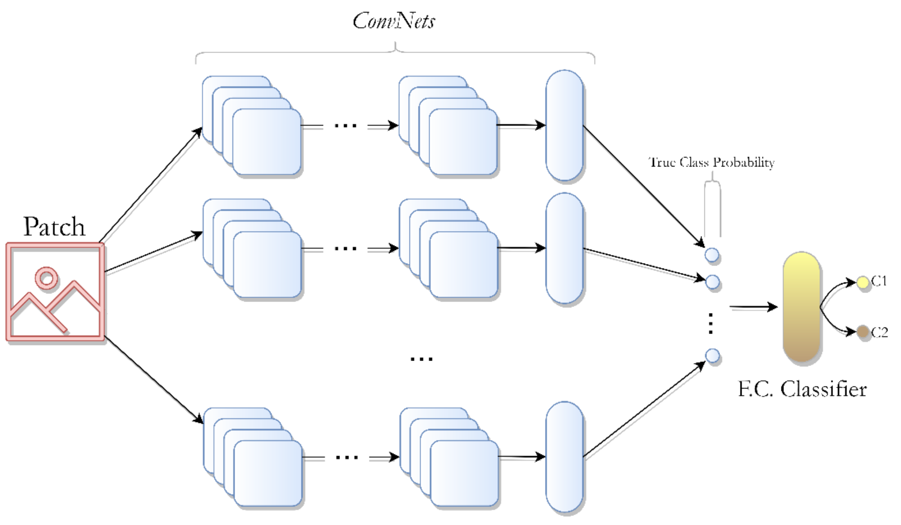


Figure 9: Stacked ensemble schema used in the project

TRAINING AND RESULTS

INTRODUCTION TO THE EXPERIMENTS

The experiments were performed in the same way and with the same parameters as those used for *Siamese networks* (Task 4), the only difference is that a higher ***learning rate*** (*10^{-3}*) was used, as otherwise the learning was too slow. *Augmentation* was not used as the number of *weights* in the network is very small.

To carry out the classification with the ***stacked model***, the predictions made by each of the models were *concatenated* in input to a *fully connected network* with two layers with a number of neurons equal to those of the sub-networks (5) and a *sigmoidal layer* of a single neuron for classification. It was chosen not to use a single layer in order not to limit the network to a linear combination of models, but to leave it free to decide the best way to perform the combination.

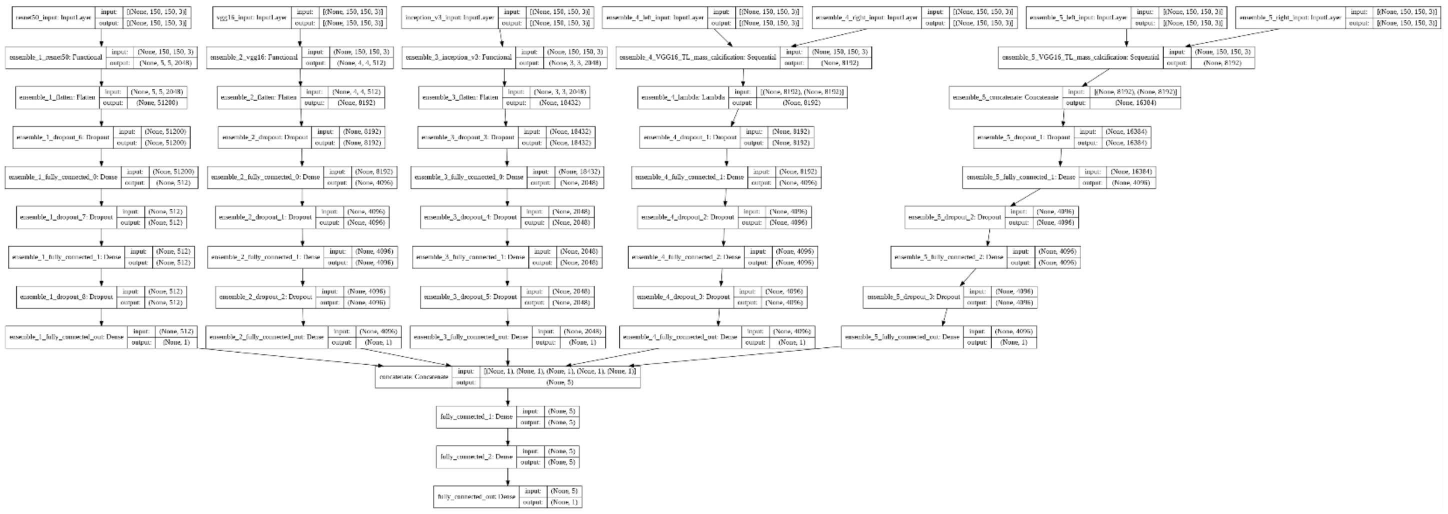
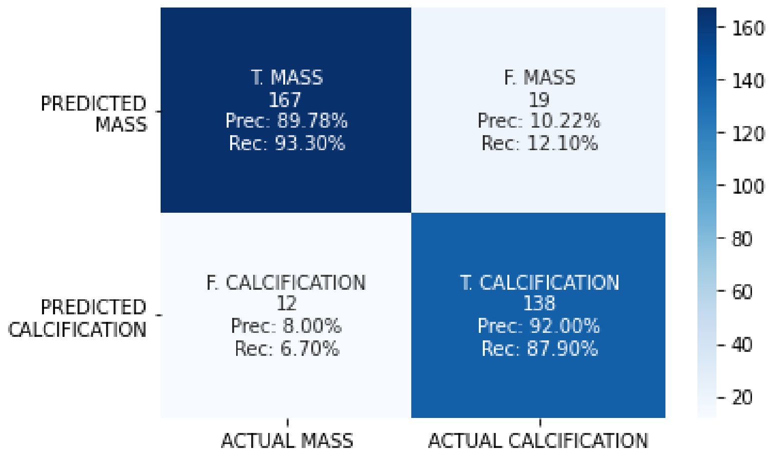


Figure 10: Mass-Calcification stacked ensemble classifier schema

MASS-CALCIFICATION CLASSIFIERS

WEIGHTED ENSEMBLE

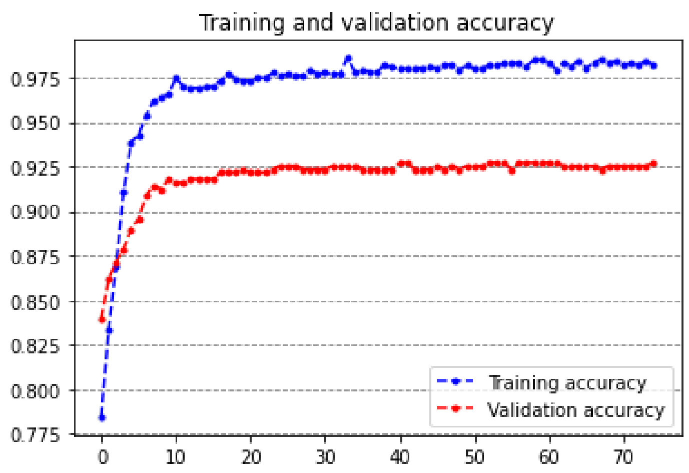
Confusion Matrix:

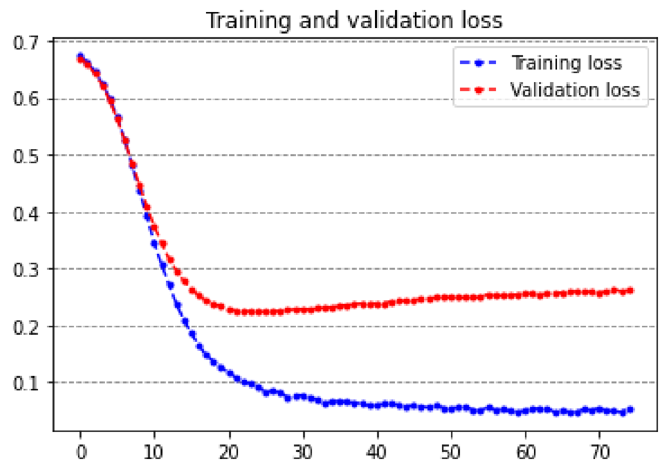


Accuracy on Training Set: *90.77%*

The simple *weighted ensemble* has better accuracy than the models taken individually, but this result can still be improved if the classifiers are better combined as shown below.

STACKED ENSEMBLE

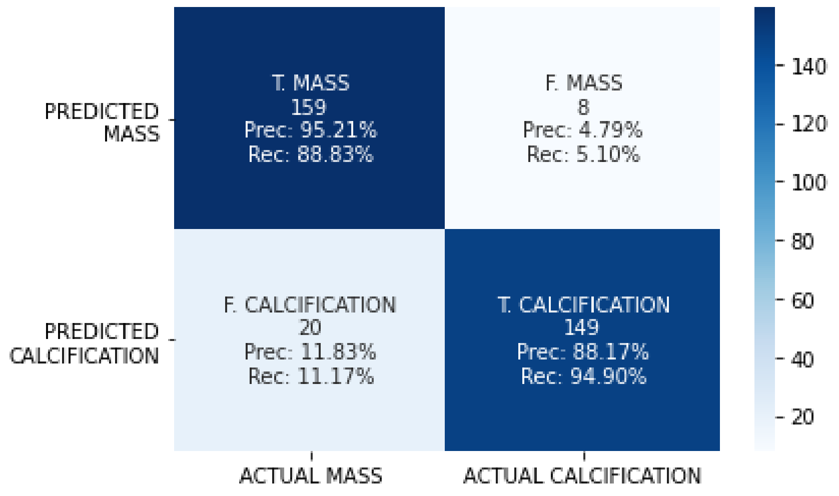




The training is very stable, even if there seems to be a strong *overfitting*. Theoretically this shouldn't happen as the training takes place on a very small network. The explanation is that the better accuracy on the training set is due to the *subnets* of which the stacked ensemble is composed. Indeed, all have better performances in training than in validation, and this is also reflected in their ensemble.

Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
97.06%	0.096	92.54%	0.2234	**91.67%**	**0.2408**

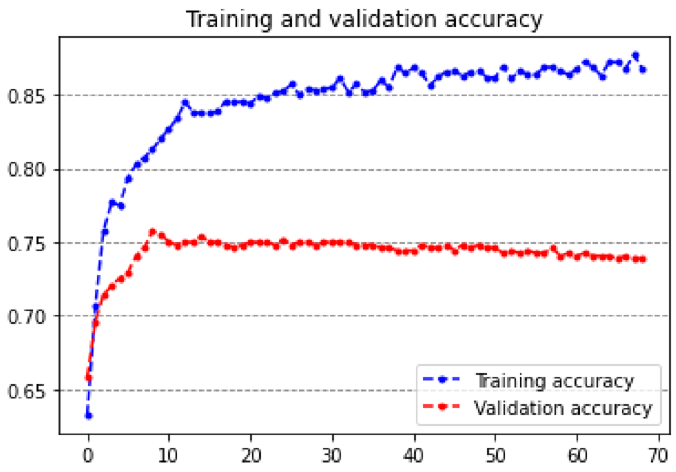
Confusion Matrix:

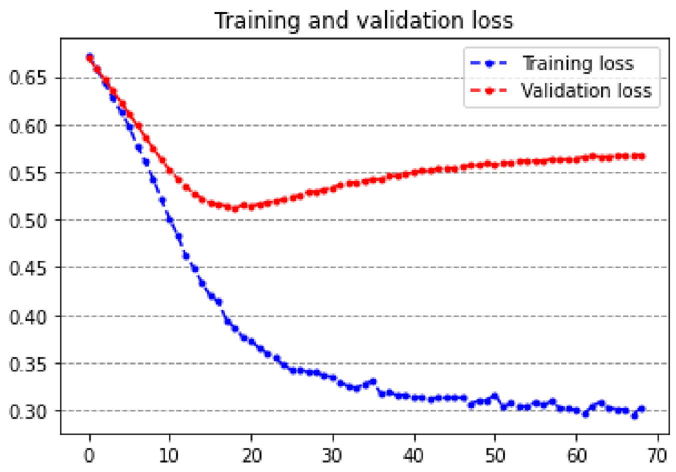


Letting the network choose the best combination almost always leads to better or equal results to the best classifier in the *ensemble*. This is reflected in the highest level of *accuracy* found to date. It should be noted that the ensemble still reflects a certain tendency of the members to prefer *calcifications* to the *masses*.

BENIGN-MALIGNANT CLASSIFIERS

STACKED ENSEMBLE

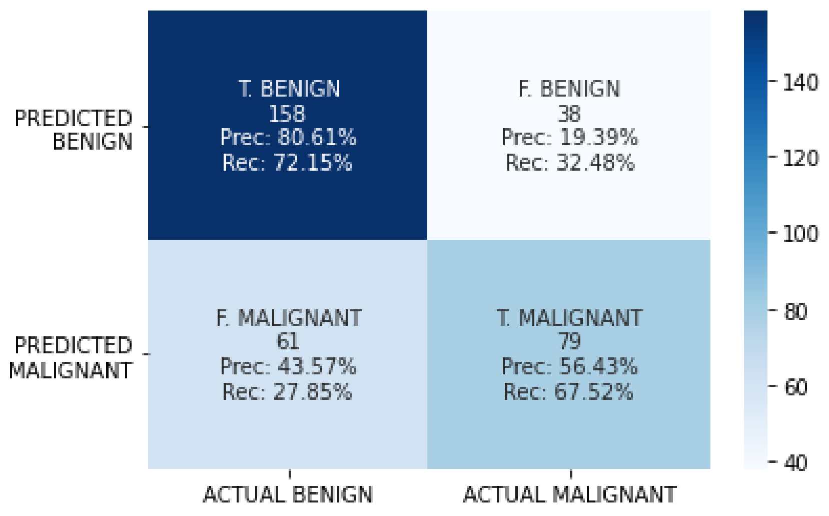




The training is stable also in this case and the reason for the apparent overfitting is the same found for *masses and calcifications*.

Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
85.09%	0.3669	74.63%	0.512	**70.54%**	**0.5887**

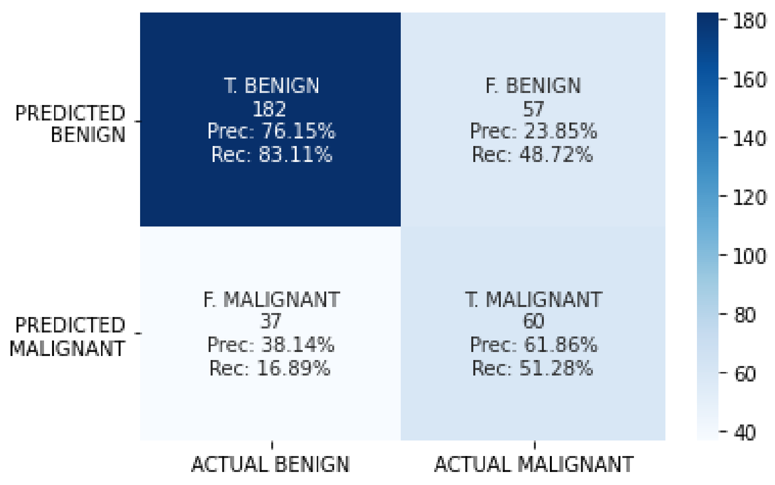
Confusion Matrix:



This is one of the cases in which the ensemble is not fruitful and therefore the results are similar to those of the best subnet: ***VGG16 FT***. it is however appreciable that there is a further step towards the *re-equilibration* of the classes, which is fundamental since the class of *malignant abnormalities* is probably the most important in a hypothetical practical application.

WEIGHTED ENSEMBLE

Confusion Matrix:

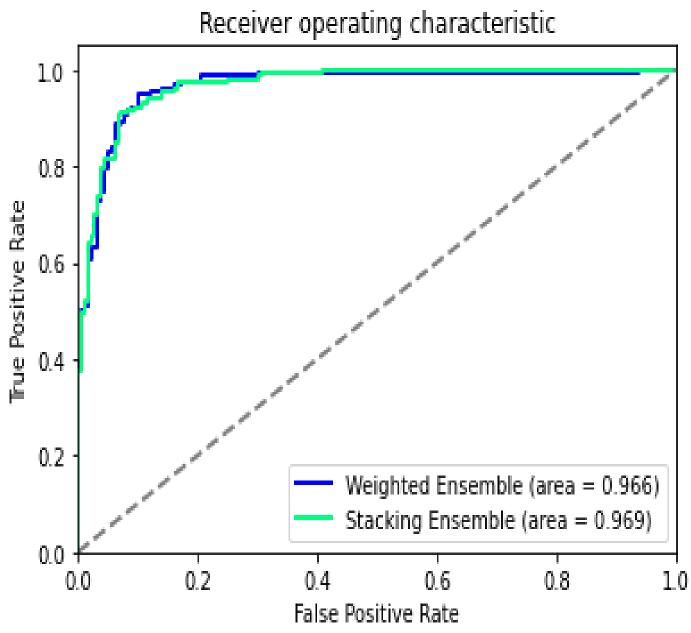


Accuracy on Training Set: *72.02%*

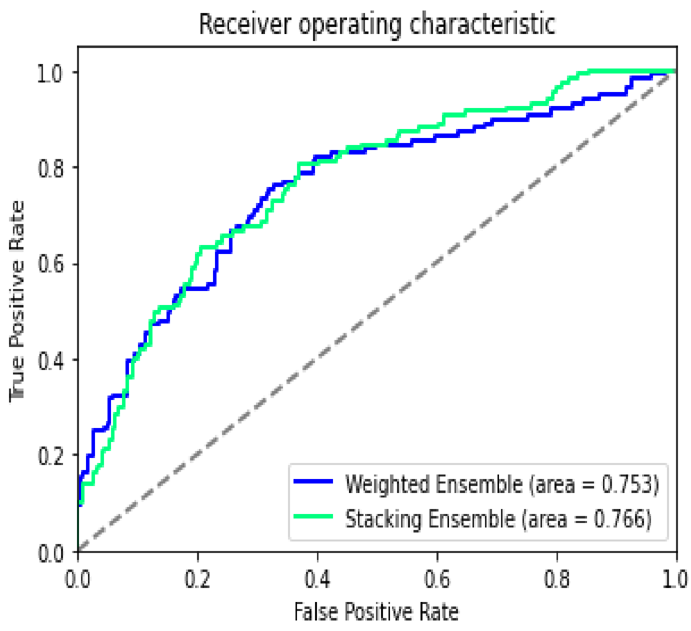
In this case the *accuracy* is better due to an *imbalance* towards the majority class.

RESULTS

Mass-Calcification Classification:



Benign-Malignant Classification:



It can be argued that the use of *weighted ensemble* methods can often lead to improvements, but only if the way in which classifiers are joined is appropriate. In fact, the union of only the two best classifiers can be better than joining three classifier or worse than taking only the best one. There is not a clear rule.

It can also be said that the ***Stacking Ensemble*** is a very safe method of aggregating subnets, which is always worth trying without the worry that adding too many networks could jeopardize the result.

Finally, it is possible to appreciate how the ***ROC curves*** resulting from the *ensemble* methods are very regular and this allows to have greater freedom in using different *thresholds* if is needed to give greater *recall* to one or the other class.

REFERENCES

PAPERS

[1] M. P. W. G. A. A. C. Lenin G. Falconi, "Transfer Learning and Fine Tuning in Breast Mammogram Abnormalities Classification on CBIS-DDSM Database," *Advances in Science, Technology and Engineering Systems Journal,* 2020.

[2] L. Tsochatzidis, L. Costaridou and I. Pratikakis, "Deep Learning for Breast Cancer Diagnosis from Mammograms—A Comparative Study," *Journal of Imaging,* 2019.

[3] L. R. M. J. H. R. E. F. R. M. W. S. Li Shen, "Deep Learning to Improve Breast Cancer Detection on Screening Mammography," *Scientific Reports - naturesearch,* 2019.

[4] O. D. X. L. M. H. Y. R. M. Richa Agarwal, "Automatic Mass Detection Mammograms Using Deep Convolutional Neural Networks," *Journal of Medical Imaging,* 2019.

[5] D. A. Ragab, M. Sharkas, S. Marshall and J. Ren, "Breast cancer detection using deep convolutional neural networks and support vector machines," *PeerJ Journals,* 2019.

[6] K. Zuiderveld, "Contrast limited adaptive histogram equalization," *Graphics gems IV,* 1994.

[7] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv,* 2016.

[8] I. Kande and M. Castelli, "The effect of batch size on the generalizability of the convolutional neuralnetworks on a histopathology dataset," *ScienceDirect,* 2020.

[9] P.-H. C. M. L. G. Q. B. C. a. G. C. Yutong Yan, "Multi-tasking Siamese Networks for Breast Mass Detection Using Dual-View Mammogram Matching," *ResearchGate,* 2020.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv,* 2014.

[11] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2016.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going deeper with convolutions," *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2015.

[13] J. Bjorck, C. Gomes, B. Selman and K. Q. Weinberger, "Understanding Batch Normalization," *Cornell University,* 2018.