Computational Intelligence and Deep Learning
Academic Year 2020/2021

Convolutional Neural Network for Medical Imaging Analysis
Abnormality detection in mammography

Autors:
Elena Veltroni
Federica Verna
Filippo Minutella

Contents

1 Introduction

This project aims to classify anomalies in mammography using convolutional neural networks.
The dataset of interest is the CBIS-DDSM (Curated Breast Imaging Subset of Digital Database for Screening Mammography) [1]: a digital breast imaging dataset performed during screening mammography. The scale of the database along with ground truth validation makes the DDSM a useful tool in the development and testing of decision support systems.
The images have been decompressed and converted to DICOM format, the standard format for the communication and management of medical imaging information and related data.
The dataset is a collection of mammographic images containing 1458 cases of calfications and 1218 cases of masses with verified pathology information by a trained mammographer. Both can be benign or malignant, and the general task of classification is to distinguish between two classes:

- Mass or calcification
- Malignant or benign.

Furthermore, several csv files are provided with a detailed description of each image, enabling another fine-grained task: abnormality diagnosis classification. It aims at distinguishing the following classes:

- Mass, Benign (with or without callback)
- Mass, Malignant
- Calcification, Benign
- Calcification, Malignant

2 Task 1: Related works

The rapid development of deep learning has stimulated much interest in its application to medical imaging problems.
Regarding mammography, the main purpose is to identify mass and calcification but, due to their variable aspect, a significant number of breast cancer cases are missed or misdiagnosed if it is only depended on the radiologists' subjective judgement.
For this reason, lots of studies are based on the use of Convolutional Neural Networks for the identification of positive signas of mammography.

Tang C., Cui X., Yu X., & Yang F.(2019) *"Five Classification of Mammography Images Based on Deep Cooperation Convolutional Neural Network"* [2] propose a Deep Cooperation CNN (DCCNN) to classify mammography images of a data set into five categories including benign calcification, benign mass, malignant calcification, malignant mass and normal breast.
The dataset is constituted by 695 normal cases from DDSM, 753 calcification cases and 891 mass cases from CBIS-DDSM. Their model achieves **91% accuracy** and **0.98 AUC** on the test set.

Shen L., Margolies L. R., Rothstein J. H., Fluder E., McBride R. & Sieh W. (2019) *"Deep Learning to Improve Breast Cancer Detection on Screening Mammography"* [3] propose a deep learning algorithm that can accurately detect breast cancer on screening mammograms using an "end-to-end" training approach in which a model to classify local image patches is pre-trained using a fully annotated dataset with ROI information. The patch classifier's weight parameters are then used to initialize the weight parameters of the whole image classifier. On the test set of digitized film mammograms from the Digital Database for Screening Mammography (CBIS-DDSM), the best single model achieved a per-image **AUC of 0.88**, and four-model averaging improved the **AUC to 0.91 (sensitivity: 86.1%, specificity: 80.1%)**.

Ridhi A., Prateek K. & Balasubramanian R. (2020) *"Deep feature–based automatic classification of mammograms"* [4] propose a deep ensemble transfer learning and neural network classifier for automatic feature extraction and classification.
Images are pre-processed beforehand, and then they are adopted to be given as input to the ensemble model proposed. The robust features extracted are optimized

into a feature vector which are further classified using the neural network (nntraintool). The network was trained and tested to separate out benign and malignant tumors, thus achieving an **accuracy of 0.88** with an **area under curve (AUC) of 0.88**.

Falconi L. G. , Perez M., Aguilar W. G. & Conci A. (2020) *"Transfer Learning and Fine Tuning in Breast Mammogram Abnormalities Classification on CBIS-DDSM Database"* [5] propose a Transfer Learning and Fine Tuning approach applied on pre-trained convolutional neural network (ConvNet) models like: Resnet, Resnext, Xception.
The algorithm extract ROI images from the mammogram image by using the binary segmentation masks provided in the CBIS-DDSM dataset, enhances contrast and using different ConvNets models classifies breast mammogram abnormalities in benign and malignant. The best classifier performance is obtained using VGG16 which achieves an **AUC value of 0.844**.

3 Dataset

Dealing with the original dataset is critical since full images are high resolution and the DICOM format is not natively supported in tf.keras.
Indeed, the dataset is provided with numpy arrays containing images and labels from training and test sets.
The steps performed on each original image are described below:

- The abnormality patch has been extracted from the original image according to the binary mask.
- A patch of healthy tissue (baseline patch) adjacent to the abnormality patch has been extracted from the original image (left, right, top or bottom - no overlap). Both abnormality patch and baseline patch have been added to the images tensor; in other words, an abnormality patch has been ignored if a related baseline patch could not be extracted.
  Indeed, the training set is composed by:

	Benign	Malignant	Total
Train Masses	620	598	1218
Train Calcification	948	510	1458
Total	1568	1108	2676

- Class labels have been assigned to the patches according to the following mapping:
  - 0: Baseline patch
  - 1: Mass, benign
  - 2: Mass, malignant
  - 3: Calcification, benign
  - 4: Calcification, malignant

4 Preprocessing

Before this data can actually be used as input in the models, some pre-processing steps are required. In order to prepare the data correctly for the various networks the following steps are performed:

1. Import the training and test data as numpy arrays.
2. When the problem is classification of anomalies only remove the labels and images of the baseline patch.
3. Merge the labels according to which classes are involved in the classification.
4. Normalize the pixel values to be in a range compatible with the chosen model. The Scratch CNN models use input in the (0,1) range, while the various pre-trained models are designed to work with images in (0,255) over 3 channels as they are trained on RGB images.
5. Split the training data into the ”training” and ”validation” sets. The validation set is only used to monitor actual performance on an independent set during training.
6. Create Keras generators to perform data augmentation on the images to be submitted to the network.

5 Task 2: CNN from Scratch

5.1 Two classes: mass and calcification

The task involves the development from scratch of a classification model that can discriminate between mammograms with masses and those with calcifications.
Since it is very difficult to predict the performance of a network in advance, the *trial and error approach* was applied to find the best performing network, which led to perform several experiments starting from a very simple network and adjusting it according to the results obtained.
A network that is too small cannot generalize well; while a model with too many parameters may learn slowly and adapt too much.
The approach used to find the right size was to start with a small model that did not fit well and then gradually increase its size until it began to generalize.
The networks can then be refined by adjusting the size of the individual layers and using *regularization* techniques (dropout, batch normalization, l1-l2 regularization).

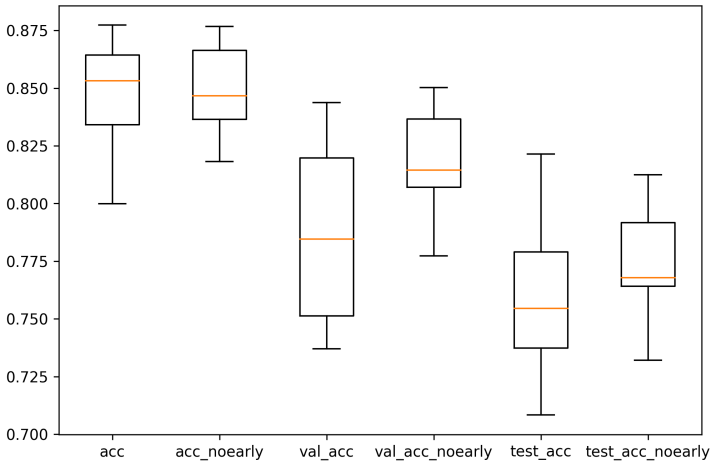
During training, the loss function on the validation set was constantly monitored for signs of overfitting, and, in order to obtain the *best model* (with its weights) generated during this phase, the one associated with the lowest validation loss is stored and used as a reference for the subsequent performance analysis. Looking at loss rather than accuracy since it is more informative: while accuracy represents only the fraction of correctly classified samples compared to the total, loss also measures the confidence the network has in making its predictions.

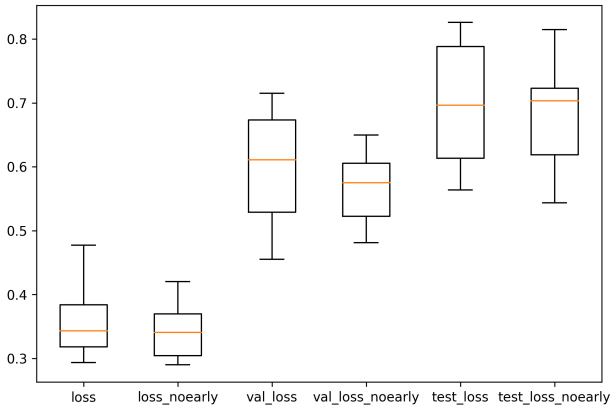
Since it is not possible to know in advance how long it will take to converge to the optimum, the network can learn for an indefinite number of times until the validation accuracy stops decreasing. This is called *early stopping*, and consists in stopping training if the network does not make progress within a certain number of epochs that can be specified by setting patience. Validation loss is a simple and effective metric for assessing the quality of a model over time, so this experiment will use it to stop training.

Dataset creation: From the original dataset with five labels, the label '0', which correspond to baseline patch, has been discarded because this task considers only the abnormalities, and label '1' and '2', which both correspond to masses, have been aggregated and label '3' and '4', which are both calcifications have been

aggregated too.
Furthermore, the original images have a depth of 16bit so they have been transformed into a float32 array and normalized between 0 and 1.
Finally, the training set of the original dataset has been splitted into a training and a validation set, while the test set is given separately.

Variability Analysis: Due to the use of GPU, it is not possible to have the same results with consequent running of one model. Hence, a variability analysis has been performed considering 12 executions of the same model and another 12 executions of the same model with Early stopping and saving the best model. From the images below, it seems that with the Early stopping and saving the best model it is possible to reduce the results variability in both validation and test.





5.1.1 Experiment 1 - Basic model

The first model is a very small CNN, consisting of 2 convolutional layers alternating with max-pooling. The last two layers are a fully connected layer: the first consisting of 16 neurons and ReLU (Rectified Linear Unit) activation and the last consisting of a single neuron with sigmoid activation that generates the final output.

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3,3), activation='relu', input_shape=(150,150,1)))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(64, (3,3), activation='relu'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Flatten())
model.add(layers.Dense(16, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

As for the optimizer, RMSprop, an adaptive algorithm proposed by G. Hinton, is used, which is considered by many to be reliable and safe in most situations. Parameters, such as learning rate and decay, are initially left with their default values. The batch size is set to 64, which is the number of samples considered in each iteration. After all iterations, the result obtained is described in the following table:

Validation accuracy	Validation loss	Testing accuracy	Testing loss
82.68	0.48	79.46	0.55

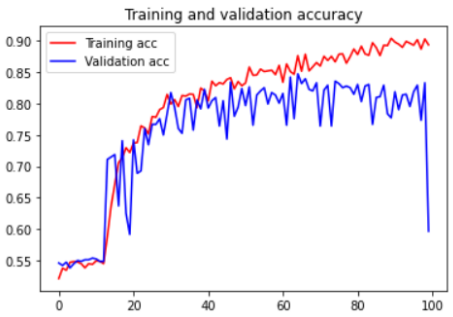


Figure 1: Accuracy

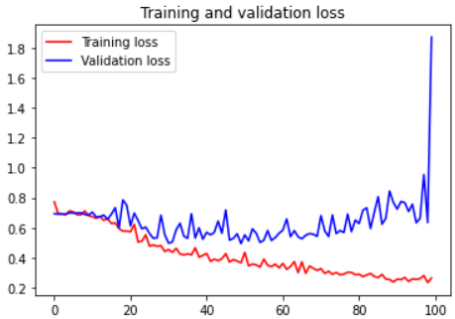


Figure 2: Loss

In the end, the network achieved an almost perfect classification on the training dataset, which of course does not make it suitable for classifying other data. In

fact, the accuracy achieved on the validation set was 82.68%, with 79.46% on the test set. In fact, the model overfits, mainly due to the limited data set (only 2000 images for 1.3 million weights) and the lack of regularization methods. Nevertheless, this is a good starting point for subsequent models.

5.1.2 Experiment 2 - Data augmentation

Given the result obtained in the previous experiment, data augmentation was added as a regularization method to eliminate or reduce overfitting.

Data Augmentation: Recent advances in deep learning models have been largely attributed to the amount and diversity of data collected in recent years. Increasing data is a strategy to significantly increase the diversity of data available for training models, without collecting new data, and thus allows for decreasing or eradicating network overfitting. This happens when the built model starts to store training data values instead of learning from them.
On the contrary, doing too much data augmentation on the training set can still bring disadvantages as we would have that the trend of the accuracy of the training set is much worse than that of the validation, because the images that the network has to recognise have been modified too much, so it is necessary to find a trade off.
The Keras deep learning neural network library provides the capability to fit models using image data augmentation via the *ImageDataGenerator* class. The data augmentation applied to the training set is taken from the paper Huang M.L. & Lin T.Y (2020) *"Dataset of breast mammography images with masses"*[6] which proposes to perform data augmentation with multi-angle rotation and then to flip the images horizontally and vertically.

```
train_datagen = ImageDataGenerator(
    rotation_range=180,
    vertical_flip = True,
    horizontal_flip = True
)
```

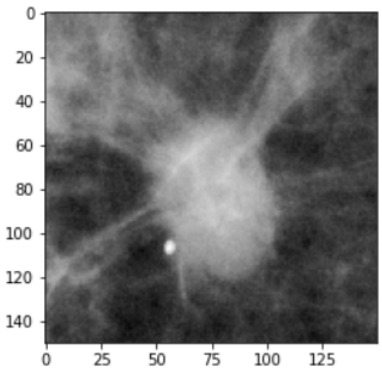


Figure 3: Original

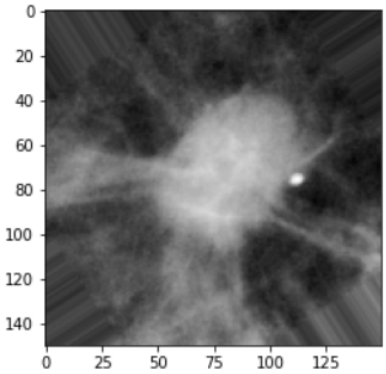


Figure 4: Rotated

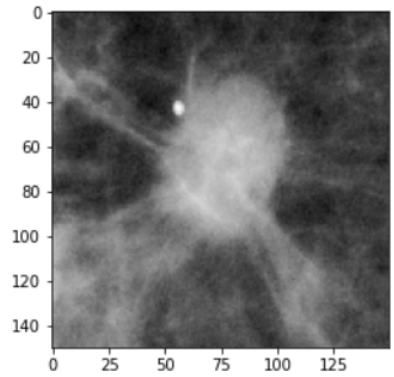


Figure 5: Vertical flip

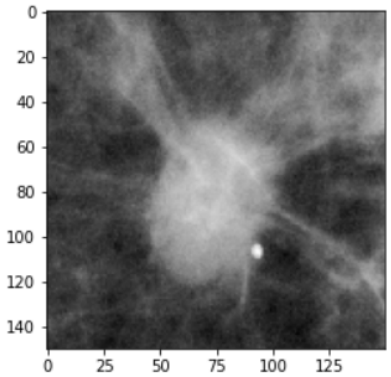


Figure 6: Horizontal flip

Thanks to data augmentation, overfitting is no longer present, as both training and validation loss decreased over time (with some noise). The accuracy slightly dropped because augmentation makes the learning task harder by design, however the network kept learning during the whole run, as the training accuracy suggests.

Validation accuracy	**Validation loss**	**Testing accuracy**	**Testing loss**
80.82	0.43	78.57	0.50

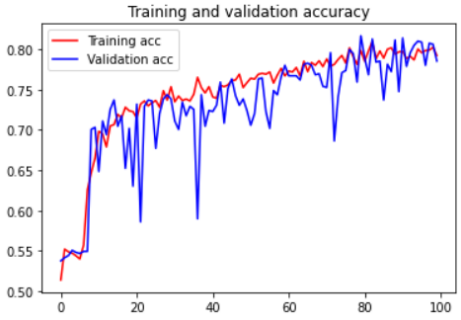


Figure 7: Accuracy

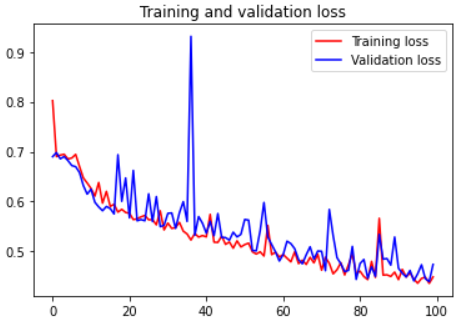


Figure 8: Loss

5.1.3 Experiment 3 - Adam optimizer

So far, all experiments have been conducted using RMSprop, which is a widely respected optimiser, but not necessarily the best choice in every situation. Another popular optimization algorithm is Adam, an extension of stochastic gradient descent. It was decided to use Adam because it can converge faster than RMSprop when the learning rate is set correctly. In this experiment, we set it to 0.0001.

Validation accuracy	Validation loss	Testing accuracy	Testing loss
78.08	0.49	75.30	0.53

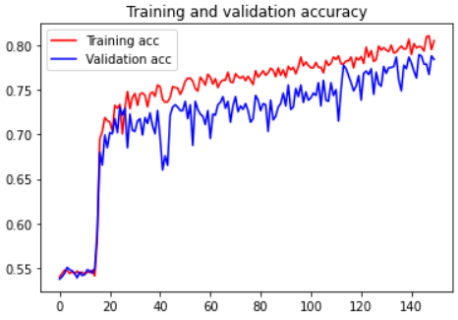


Figure 9: Accuracy

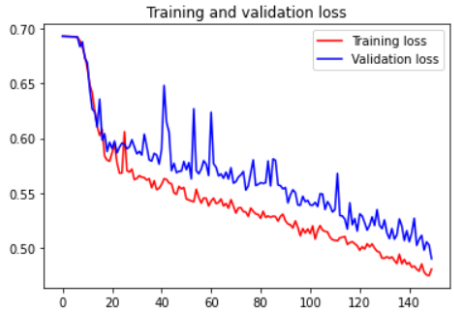


Figure 10: Loss

As can be seen from the results, although the accuracy on both the validation and test set has decreased slightly, thanks to the Adam optimizer the accuracy and leakage trends are more stable and have less fluctuation.

5.1.4 Experiment 4 - Batch normalization

This experiment aims to test the effects of batch normalization, a popular technique used to improve the speed, performance and stability of a network. It consists in normalizing (subtract mean and divide by variance) the inputs of each layer for each mini-batch. The optimal position of batch normalization w.r.t. the activation layer is still debated: although the paper authors used to put it before the activation to avoid altering the output distribution, many researchers have obtained better results by placing it after the Relu [8][9]. The final layer is usually excluded from batch normalization, as non-normalized inputs typically help predicting the correct class.

```
model4 = models.Sequential()
model4.add(layers.Conv2D(32, (3,3), activation='relu', input_shape=(150,150,1)))
model4.add(layers.BatchNormalization())
model4.add(layers.MaxPooling2D((2,2)))
model4.add(layers.Conv2D(64, (3,3), activation='relu'))
model4.add(layers.BatchNormalization())
model4.add(layers.MaxPooling2D((2,2)))
model4.add(layers.Flatten())
model4.add(layers.Dense(16, activation='relu'))
model4.add(layers.Dense(1, activation='sigmoid'))
```

Validation accuracy	Validation loss	Testing accuracy	Testing loss
86.05	0.33	84.22	0.37

It is easy to see that batch normalization causes wild oscillations in validation loss and accuracy.

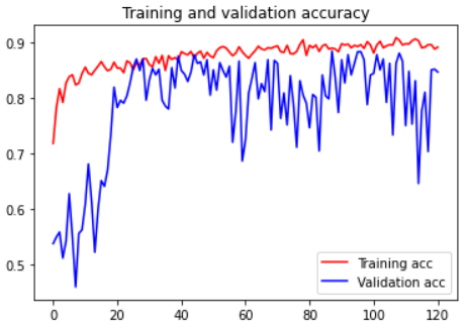


Figure 11: Accuracy

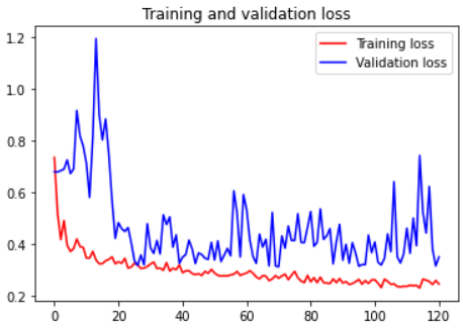


Figure 12: Loss

5.1.5 Experiment 5 - L2 regularization

From the previous result, obtained from inserting batch normalization into the model, an accuracy and loss trend is obtained with many fluctuations that are perhaps due to slight overfitting, so this experiment attempts to remove it by adding l2 regularization to each convolutional layer.

```
model5 = models.Sequential()
model5.add(layers.Conv2D(32, (3,3), activation='relu', input_shape=(150,150,1),
    kernel_regularizer=l2(1)))
model5.add(layers.BatchNormalization())
model5.add(layers.MaxPooling2D((2,2)))
model5.add(layers.Conv2D(64, (3,3), activation='relu', kernel_regularizer=l2(1)))
model5.add(layers.BatchNormalization())
model5.add(layers.MaxPooling2D((2,2)))
model5.add(layers.Flatten())
model5.add(layers.Dense(16, activation='relu'))
model5.add(layers.Dense(1, activation='sigmoid'))
model5.summary()
```

Validation accuracy	Validation loss	Testing accuracy	Testing loss
88.915	0.66	87.20	0.71

The L2 regularisation managed to remove the overfitting clearly visible in the graphs below, so that the validation loss finally corresponds to the training loss without deviations; unfortunately, however, the accuracy trend still has many fluctuations.

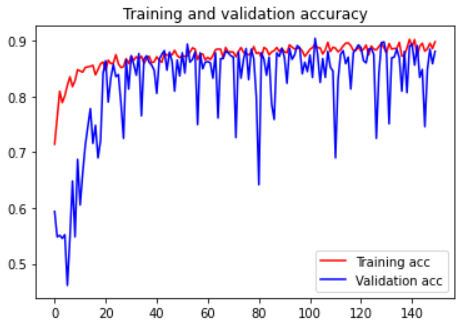


Figure 13: Accuracy

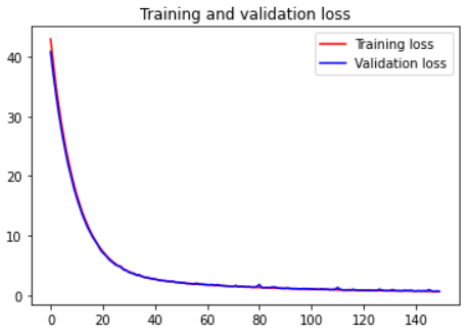


Figure 14: Loss

5.1.6 Other experiments

In subsequent experiments the number of convolutional layers, fully connected layers and the number of neurons in the latter were changed.

- The Model 6 evolves from Model 3 by simply increasing to 64 the number of neurons in the fully-connected layer.

Validation accuracy	Validation loss	Testing accuracy	Testing loss
80.19	0.43	77.08	0.48

- Model 7 is like the previous one, with the addition of an extra convolutional block with 128 features. The purpose of this experiment is to verify if a deeper convolutional stage helps improving the accuracy or not. This strongly depends on the nature of the problem; in general, it is strongly related to the kind of object to classify.

Validation accuracy	Validation loss	Testing accuracy	Testing loss
89.53	0.29	86.31	0.34

- As increasing the number of convolutional layers brought benefits in terms of accuracy and loss, an additional convolutional layer with 256 features is added in Model 8.

Validation accuracy	Validation loss	Testing accuracy	Testing loss
89.04	0.27	86.01	0.34

- Model 9 attempts to make the network even deeper by adding an additional convolutional layer with 256 featers.

Validation accuracy	Validation loss	Testing accuracy	Testing loss
89.41	0.26	84.23	0.35

- Since in the last experiment the accuracy decreased a lot, perhaps due to the fact that by adding too many convolutional layers we decreased too much the number of parameters, let's restart from Model 8 by adding in this experiment, in Model 10, a fully connected layer composed of 256 neurons while we change the other one from 64 to 128. In this way it is possible to obtain about 3 million parameters.

Validation accuracy	Validation loss	Testing accuracy	Testing loss
90.28	0.25	87.20	0.31

- At this point, using Model 11, an attempt is made to stabilize the network by alternating dense layers with drop out layers at which a rate of 0.3 has been set.

Validation accuracy	Validation loss	Testing accuracy	Testing loss
89.29	0.25	85.12	0.32

5.1.7 Experiment 12 - Contrast enhancing

To try to obtain a better result, the contrast of the images has been increased to see if with this modification, the best model drawn so far, is able to better recognize the anomalies present in the dataset.

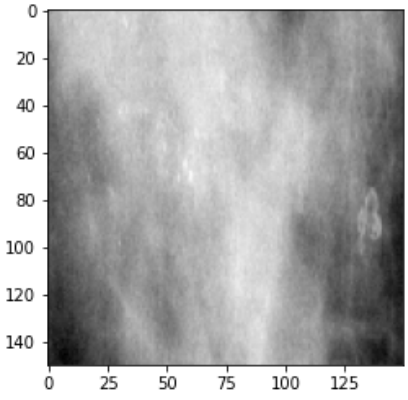


Figure 15: Original

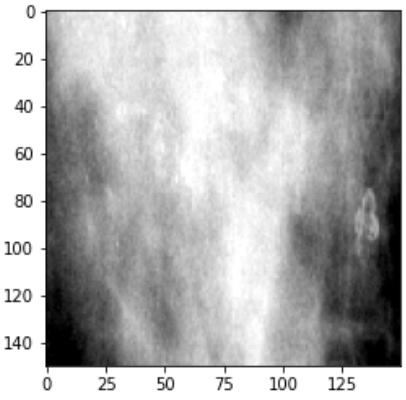


Figure 16: Contrast enhanced

```
def contrast_stretching(img):
    p2, p98 = np.percentile(img, (2,98))
    img_modified = exposure.rescale_intensity(img, in_range=(p2, p98))
    return img_modified
```

With this function, an analysis is made of the distribution of pixel densities in an image and then the image is rescaled to include all intensities that fall within the 2nd and 98th percentiles.

Validation accuracy	Validation loss	Testing accuracy	Testing loss
87.67	0.28	85.12	0.36

As the graph shows, there is a clear improvement in the trend of both precision and loss functions, in fact not only is more stable but the model is able to learn very quickly, and then improve in all epochs, unfortunately losing some precision in the test, probably by increasing the number of epochs and changing some parameters in the model could be found a very good classifier.

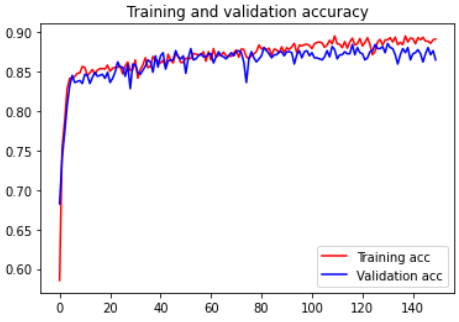


Figure 17: Accuracy

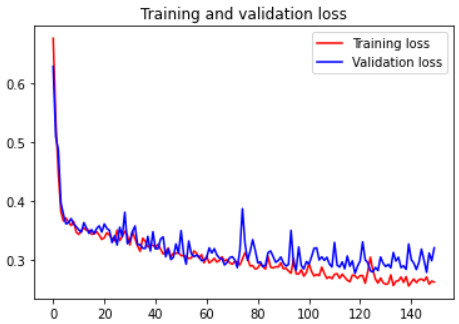


Figure 18: Loss

5.1.8 Error analysis

Let's take a look at the results obtained from the previous experiments, analyzing in particular the samples misclassified by the best model obtained, Model 10 (see section 5.1.6).
An accuracy of approximately 87% and an AUC of 0.95 is obtained with this network.

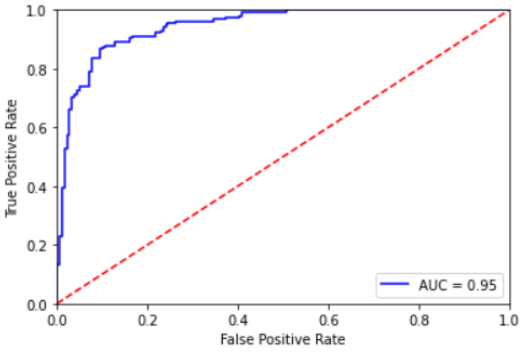


Figure 19: AUC

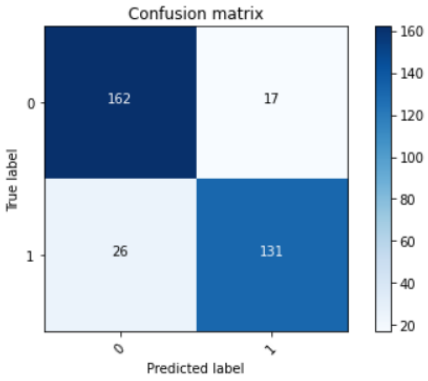


Figure 20: Confusion matrix

As can be seen from the confusion matrix, the model misses only 43 out of 336 images:

Number of test images	336
Mispredictions	43
Masses classified as Calcifications	17
Calcifications classified as Masses	26

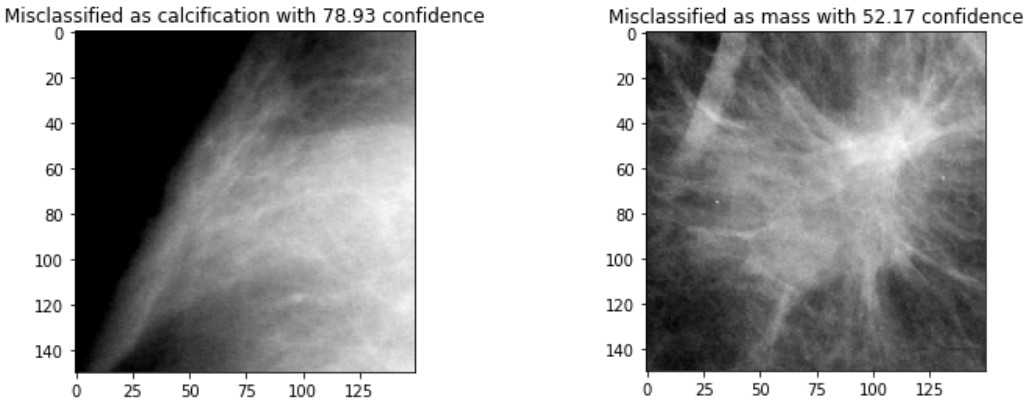
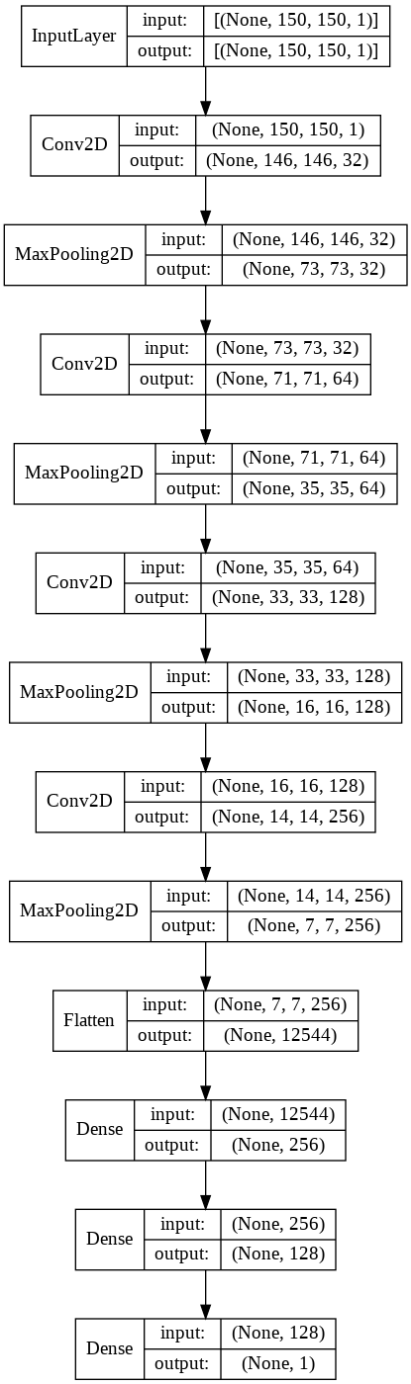


Figure 21: Example of misclassified images

5.1.9 Architecture of the best model



5.2 Two classes: benign and malignant

The task is similar to the previous one but, instead of distinguishing between masses and calcifications, it concerns the diagnosis of abnormality, i.e. whether it is benign or malignant.

Dataset creation: As in the previous task, described in section 5.1, the labels equal to 1 and their respective images have been removed since they do not have any abnormalities within it, while the labels '1' and '3', that both correspond to benign abnormalities, and '2' and '4', that instead identify malignancies, have been combined under a single class.
As done in the previous task, the original images have been transformed into a float32 array and normalized between 0 and 1 and then a training and a validation set have been created.

The dataset is a little bit imbalanced, particularly the training set has 1097 elements from class "benign" and 776 elements from class "malignant". For this reason, a function in the fitting model has been introduced in order to assign different weights to the classes and to rebalance the datasets.

```
class_weights = class_weight.compute_class_weight('balanced', np.unique(
    train_labels), train_labels)
history = model.fit(
    ...
    class_weight = dict(enumerate(class_weights)),
    ...
)
```

Evaluation Metrics for Imbalanced Classification: Considering the type of classification, it should be noted that misclassifications do not carry the same weight, as a misclassification of a 'malignant' image is more serious than a 'benign' one.
For this reason, it has been decided to evaluate the performance of the following networks not considering the accuracy value but the F2 score value, which weights recall higher than precision. This makes this metric more suitable for our problem where it's more important to classify correctly as many positive samples as possible, rather than maximizing the number of correct classifications.

It has also been considered the area under the ROC curve as a measure of diagnostic accuracy. For the interpretation of its values, it is possible to refer to the classification proposed by Swets[7]:

1. AUC=0.5 the test is not informative;
2. $0.5 < AUC \leq 0.7$ the test is not very accurate;
3. $0.7 < AUC \leq 0.9$ the test is moderately accurate;
4. $0.9 < AUC < 1.0$ the test is highly accurate;
5. AUC=1 perfect test.

5.2.1 Experiment 1 - basic model

In order to find the model that best fits this classification problem, since it is very different from the previous one, the same *trial and error approach* has been used starting from a very simple network and changing its architecture as we went along.
The first model is a very small CNN, consisting of 2 convolutional layers alternating with max-pooling. The last two layers are a fully connected layer: the first consisting of 16 neurons and ReLU (Rectified Linear Unit) activation and the last consisting of a single neuron with sigmoid activation that generates the final output.
As for the optimizer, RMSprop has been used leaving parameters at their default values.

f2 score	**Validation accuracy**	**Validation loss**	**Test accuracy**	**Test loss**
0.38	67.32	0.64	64.88	0.64

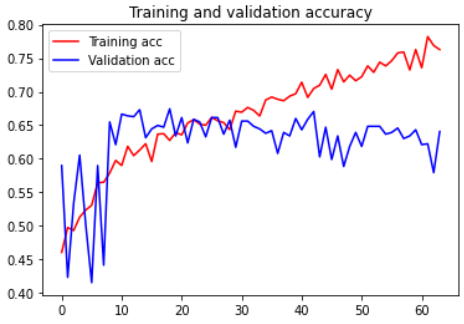


Figure 22: Accuracy

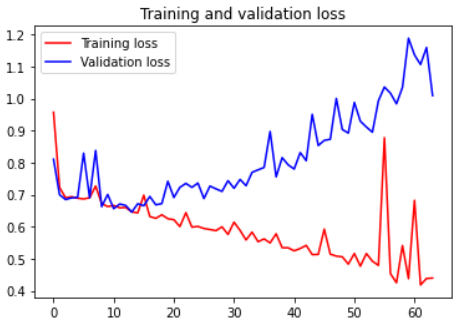


Figure 23: Loss

5.2.2 Experiment 2 - Data augmentation

As can be seen from the graphs of the result of the previous experiment, the base model overfitts. In order to try to eliminate this phenomenon, data augmentation has been introduced again with the same characteristics of the previous task (see section 5.1.2)

f2 score	Validation accuracy	Validation loss	Test accuracy	Test loss
0.58	67.06	0.58	61.01	0.64

Malignant classified as Benign	44

Although a few points in accuracy have been lost, as it can been seen, the f2 score value has greatly been increased, which indicates that this model can predict the class of malignant abnormalities much better than in the previous experiment. In addition, the accuracy and loss trends have improved and thanks to data augmentation there is no more overfitting.

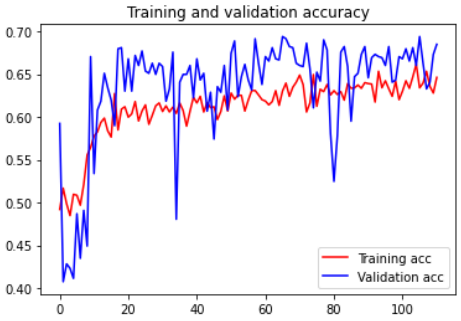


Figure 24: Accuracy

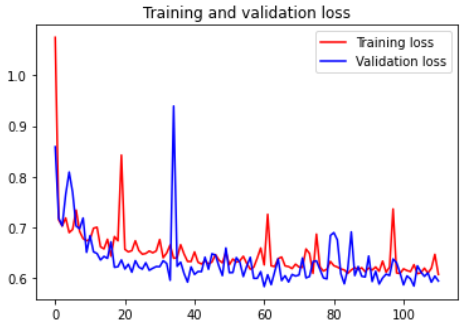


Figure 25: Loss

5.2.3 Experiment 3 - Adam optimizer

Since the trends have many fluctuations, in Model 3 Adam has been introduced as an optimizer with a learning rate equal to 0.0001.

f2 score	Validation accuracy	Validation loss	Test accuracy	Test loss
0.59	68.49	0.58	61.01	0.65

Malignant classified as Benign	41

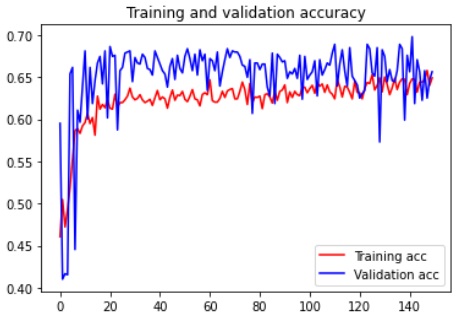


Figure 26: Accuracy

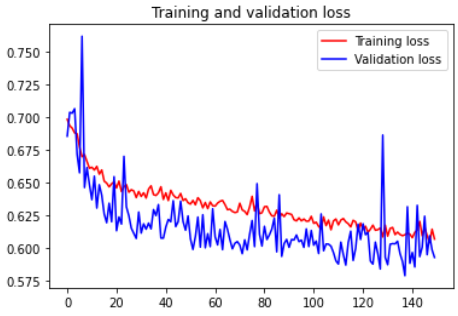


Figure 27: Loss

5.2.4 Other experiment

In subsequent experiments the number of convolutional layers, fully connected layers and the number of neurons in the latter were changed.

- Model 4 is like the previous one, with the addition of an extra convolutional block with 128 features.

f2 score	Validation accuracy	Validation loss	Test accuracy	Test loss
0.55	70.18	0.56	61.60	0.63

Malignant classified as Benign	48

- In Model 5 it has been tried to make the network deeper by adding another layer with 128 features.

f2 score	Validation accuracy	Validation loss	Test accuracy	Test loss
0.59	69.40	0.55	63.69	0.62

Malignant classified as Benign	48

- In Model 6 it has been tried to make the network even deeper by adding an additional convolutional layer with 256 features.

f2 score	Validation accuracy	Validation loss	Test accuracy	Test loss
0.62	70.57	0.54	65.48	0.62

Malignant classified as Benign	39

- Starting again from experiment 5, Model 7 has been created to which the number of neurons in the penultimate fully connected layer has been changed by doubling it to 128.

f2 score	Validation accuracy	Validation loss	Test accuracy	Test loss
0.66	68.88	0.55	62.20	0.63

Malignant classified as Benign	31

- In order to try to regularize the course of the accuracy and the loss in the Model 8, a dropout layer has been added with the rate equal to 0.3 between the two dense layers.

f2 score	Validation accuracy	Validation loss	Test accuracy	Test loss
0.66	69.66	0.56	62.50	0.63

Malignant classified as Benign	44

5.2.5 Experiment 9 - Increasing contrast

As done in the previous task, a preprocessing step has been added to increase the contrast of the images and the best model presented before has been used.

f2 score	Validation accuracy	Validation loss	Test accuracy	Test loss
0.42	69.79	0.56	67.55	0.59

Malignant classified as Benign	44

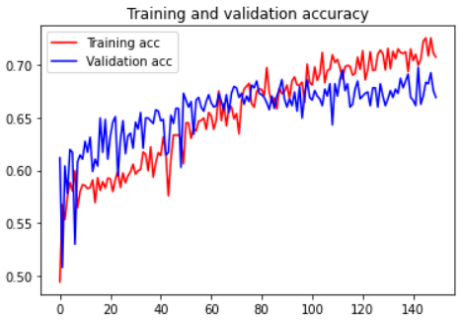


Figure 28: Accuracy

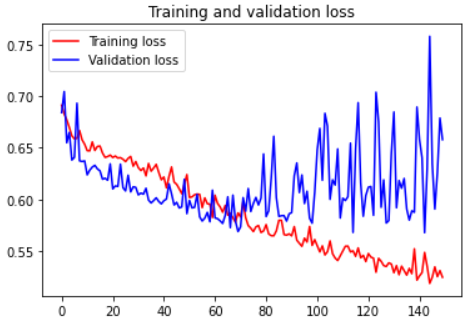


Figure 29: Loss

5.2.6 Error analysis

Let's take a look at the results obtained from the previous experiments, analyzing in particular the samples misclassified by the best model obtained, Model 7 (see section 5.2.4).
With that network, it has been managed to obtain an accuracy of about 62%, an AUC of 0.71 and an f2 score value equal to 0.66 (the highest obtained in all experiments and therefore considered the best model).
In this task the main interest is in misclassifications of malignant abnormalities as it is much more serious to get this type of class wrong.

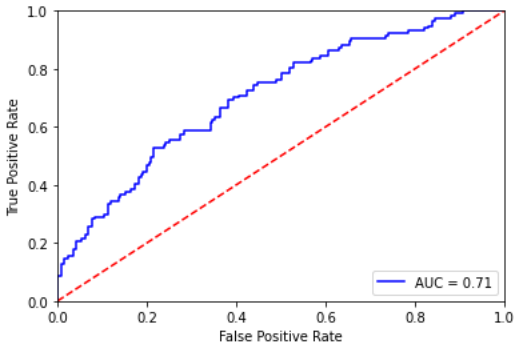


Figure 30: AUC

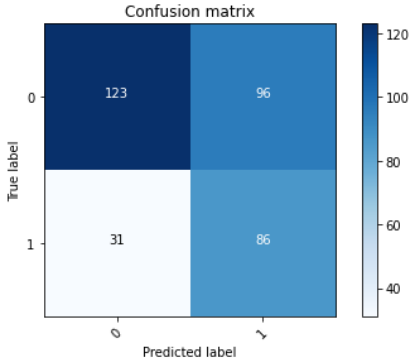


Figure 31: Confusion matrix

As can be seen from the confusion matrix, the model misses only 127 out of 336 images, but only 31 malignant classified as benign:

Number of test images	336
Mispredictions	127
Malignant classified as Benign	31

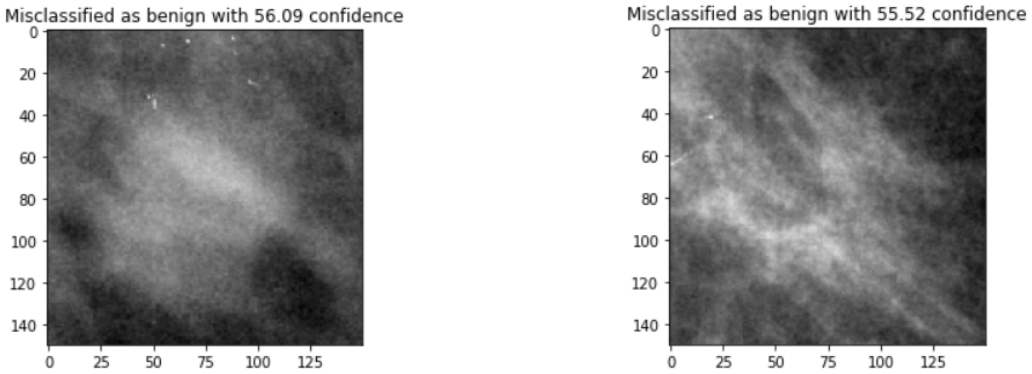
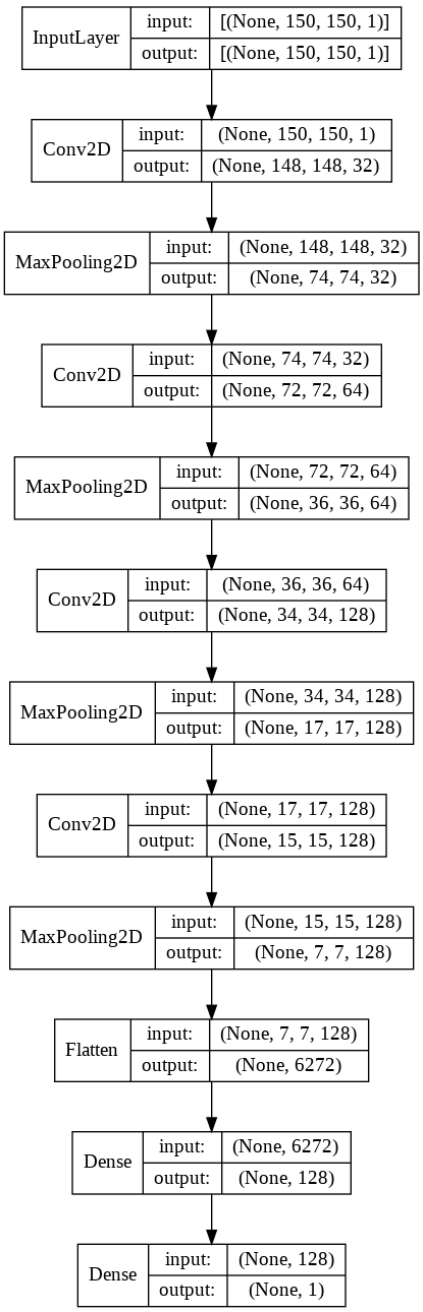


Figure 32: Example of misclassified images

5.2.7 Architecture of the best model



6 Task 3: CNN using a Pretrained network

Image classification has been accelerated by the advent of Transfer Learning, which allows to use a pre-existing model, trained on a huge dataset and developed for a task, as the starting point for a model on a second task. Consequently, by reducing the cost of training new deep learning models and because the datasets have been checked, the quality can be assured.

6.1 Introduction

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper *"Very Deep Convolutional Networks for Large-Scale Image Recognition"*.
VGG16 has several 3x3 convolutional layers in cascade occasionally interleaved with 2x2 maxpooling layers. It's able to achieves 90.1% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. Its enormous size makes the training an extremely slow process; nevertheless, VGG16 is often used for transfer learning, thanks to its flexibility.

ResNet50 is a short name for Residual Network and, as the name of the network indicates, it introduces residual learning where instead of trying to learn some features, the network tries to learn some residual that are subtractions of feature learned from input of that layer.
The ResNet50, 50 layer Residual Network, achieves 92.1% top-5 test accuracy in ImageNet.

InceptionV3 is a convolutional neural network for assisting in image analysis and object detection, and got its start as a module for Googlenet. Inception Modules are used in Convolutional Neural Networks to allow for more efficient computation and deeper Networks through a dimensionality reduction with stacked 1×1 convolutions. The modules were designed to solve the problem of computational expense, as well as overfitting, among other issues. The solution is to take multiple kernel filter sizes within the CNN, and rather than stacking them sequentially, ordering them to operate on the same level.
The model achieves 93.7% top-5 test accuracy in ImageNet.

Dataset creation: Starting from the original dataset, the label '0', i.e., the one corresponding to the baseline patch, was discarded while the labels '1' and '3', which both correspond to benign abnormality was aggregated, and also the labels '2' and '4', which are both malignant. Then, the images were transformed into a float32 array and normalized between 0 and 1 (see section 5.2).

Data Augmentation: As seen in the previous paragraphs inserting data augmentation brings benefits, so it has been used also for this task since the first experiment continuing to be based on the paper Huang M.L. & Lin T.Y (2020) *"Dataset of breast mammography images with masses"*[6].

In the following tasks, all the pretrained network described above have been used and the best model has been choosen.

6.2 VGG16 mass and calcification

6.2.1 Features extraction

Feature extraction is a transfer learning approach that involves taking a model already trained on another (possibly similar) dataset, cutting off the final fully connected layer, and replacing it with a new one. The convolutional basis is frozen, which is made untrainable because it serves as a feature extractor for the new images and should not be altered by the training process.

Vanilla
The original VGG16 comes with a couple of 4096 FC layers followed by 1000 softmax neurons, which is alright for ImageNet but definitely oversized for our purpose. Hence, the convolutional base is left as it is, and the fully-connect block is shrinked to a single layer with 512 neurons feeding one sigmoidal output unit.

Validation accuracy	Validation loss	Test accuracy	Test loss
89.84	0.27	81.85	0.41

The network learns very fast to distinguish between masses and calcifications, with an accuracy of 82%.

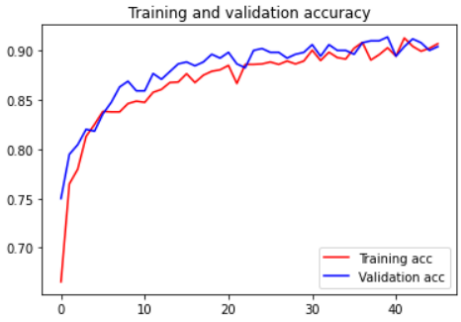


Figure 33: Accuracy

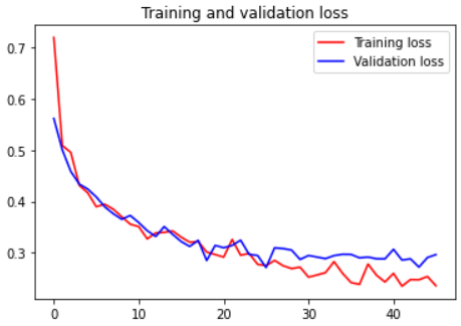


Figure 34: Loss

Smaller FC layer

The fully connected layer with 512 layers is replaced with a layer with 256 neurons.

Validation accuracy	Validation loss	Test accuracy	Test loss
91.02	0.25	83.63	0.38

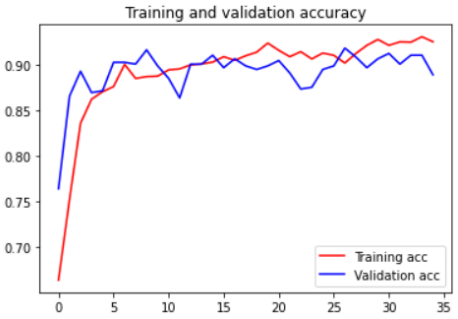


Figure 35: Accuracy

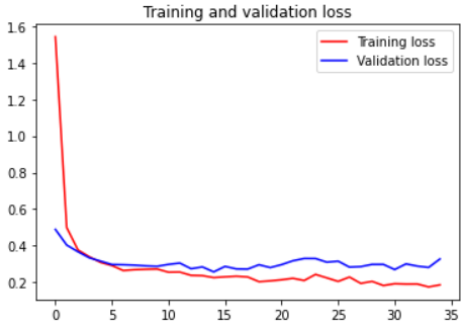


Figure 36: Loss

Two FC layers

In this experiment two fully connected layers are inserted, the former with 512 neurons and the latter with 256.

Validation accuracy	Validation loss	Test accuracy	Test loss
90.82	0.27	85.71	0.34

The network performs very well achieving an accuracy of about 86%. On the other hand, by looking at the loss graph, it begins overfitting very soon: some regularization method is very needed.

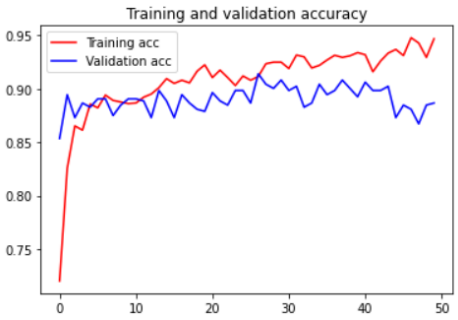


Figure 37: Accuracy

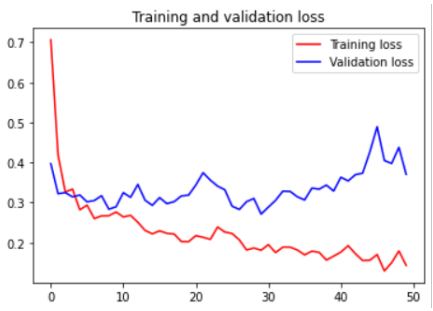


Figure 38: Loss

With dropout
Dropout is added at the end of the network to alleviate overfitting.

Validation accuracy	Validation loss	Test accuracy	Test loss
90.04	0.25	85.12	0.36

As expected, dropout mitigated the magnitude of overfitting, but it was not enough to eliminate it.

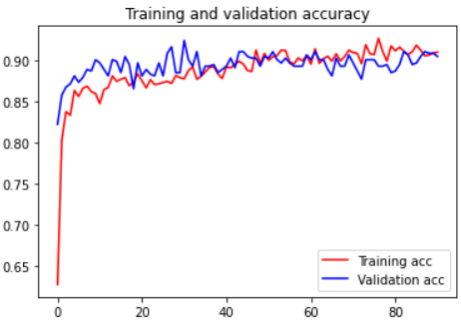


Figure 39: Accuracy

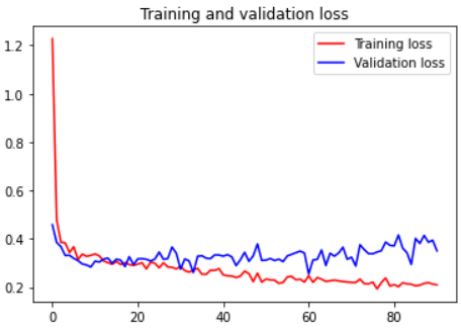


Figure 40: Loss

6.2.2 Fine tuning

Fine tuning extends feature extraction, unfreezing (i.e. making trainable) one or more of the last convolutional layers after the new FC block has been trained. Training for few additional epochs these convolutional layers get a change to be slightly adjusted (fine-tuned) to boost performance.

One layer
Starting from the model 4, the 3rd Conv2D layer in the 5th block is unfrozen and the network trained.

Validation accuracy	Validation loss	Test accuracy	Test loss
91.41	0.24	88.39	0.32

Three layers
The same process can be applied iteratively, this time starting from the 1-layer finetuned model and unfreezing the last three convolutional layers.

Validation accuracy	Validation loss	Test accuracy	Test loss
92.97	0.18	89.29	0.32

Thanks to fine tuning, the NN scored 89.29% accuracy on the testing set, with a gain of 4% compared to the previous feature extraction VGG16 models.

6.3 ResNet50 mass and calcification

6.3.1 Features extraction

As with VGG16, the last few dense layers are removed from ResNet50 and while all other layers are freezed. Also with this pre-trained network some experiments were tried where the number of dense layers and the number of neurons of the additional layers are changed.

Vanilla
A plain ResNet50 with two final FC layers, the former with 512 neurons and the latter with a sigmoidal neuron.

Validation accuracy	Validation loss	Test accuracy	Test loss
89.45	0.28	86.61	0.32

Smaller FC layer
The fully connected layer with 512 layers is replaced with a layer with 256 neurons.

Validation accuracy	Validation loss	Test accuracy	Test loss
92.38	0.21	87.50	0.31

Two FC layers
The fully connected layers become 2 one with 512 neurons and the other with 256.

Validation accuracy	Validation loss	Test accuracy	Test loss
93.16	0.20	88.39	0.35

The network learns very quickly to distinguish between masses and calcifications, with about 88% accuracy and a very stable trend,

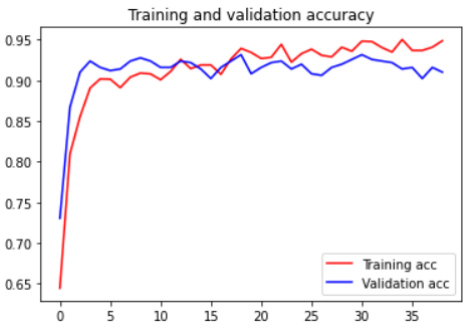


Figure 41: Accuracy

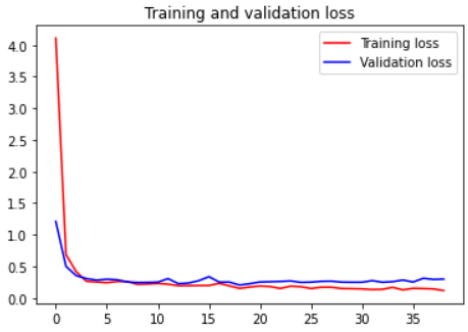


Figure 42: Loss

6.3.2 Fine tuning

One layer
Starting from the model 3, the 5th Conv2D layer in the 3rd block is unfrozen and

the network trained.

Validation accuracy	Validation loss	Test accuracy	Test loss
93.16	0.22	88.99	0.34

N layers
Starting from the best model obtained in the previous step, the last 10 convolutional layer is unfrozen and the network trained.

Validation accuracy	Validation loss	Test accuracy	Test loss
93.95	0.18	88.09	0.34

6.4 InceptionV3 mass and calcification

The same steps done with the other networks are also done with InceptionV3

Vanilla
A plain InceptionV3 with two final FC layers, the former with 512 neurons and the latter with a sigmoidal neuron.

Validation accuracy	Validation loss	Test accuracy	Test loss
89.45	0.28	88.98	0.28

Smaller FC layer
The fully connected layer with 512 layers is replaced with a layer with 256 neurons.

Validation accuracy	Validation loss	Test accuracy	Test loss
89.26	0.26	88.69	0.28

Two FC layers
The fully connected layers become 2 one with 512 neurons and the other with 256.

Validation accuracy	Validation loss	Test accuracy	Test loss
89.06	0.27	90.77	0.26

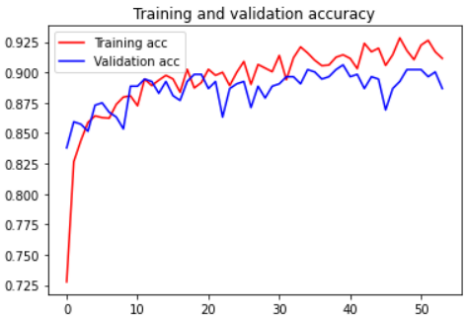


Figure 43: Accuracy

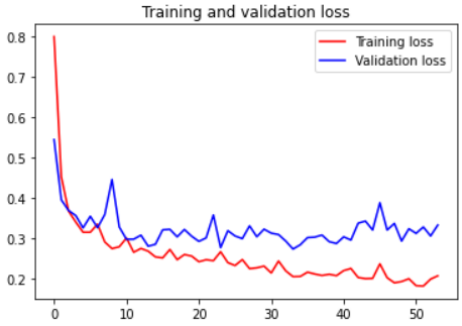


Figure 44: Loss

6.4.1 Fine tuning

One layer
Starting from the model 3, the last Conv2D layer is unfrozen and the network trained.

Validation accuracy	Validation loss	Test accuracy	Test loss
90.04	0.25	90.77	0.25

Since fine tuning did not bring any benefit, the accuracy is equal to the best model, no other experiment was implemented.

6.5 Error analysis

Several experiments were carried out for each pre-trained network to which only some FC and Dropout layers were added. The best result is obtained from the 3rd experiment performed with the pre-trained network InceptionV3 by adding 2 FC

layers with 512 and 256 neurons respectively achieving an accuracy of 91% and an AUC of 0.96.

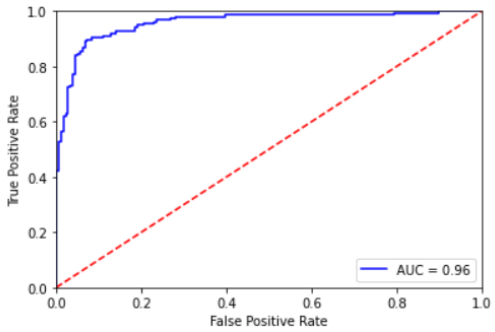


Figure 45: AUC

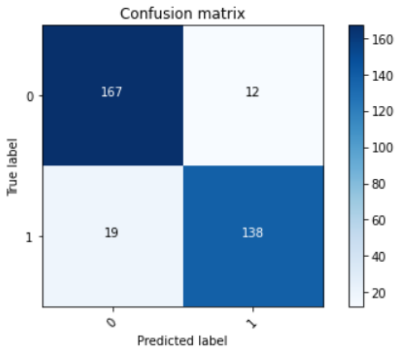


Figure 46: Confusion matrix

As can be seen from the confusion matrix, the model misses only 31 out of 336 images:

Number of test images	336
Mispredictions	31
Masses classified as Calcifications	12
Calcifications classified as Masses	19

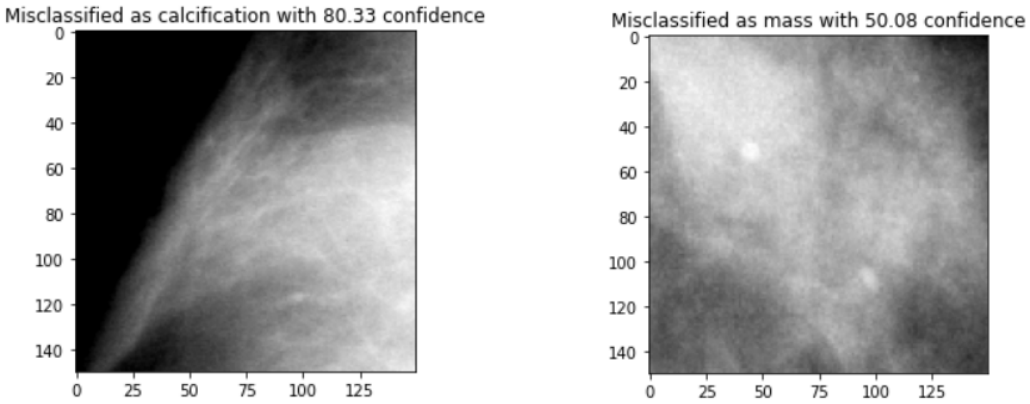


Figure 47: Examples of misclassifications

6.6 VGG16 benign and malignant

6.6.1 Features extraction

Vanilla

A simple VGG16 with two final FC layers, the first with 512 neurons and the second with one sigmoidal neuron.

f2 score	Validation accuracy	Validation loss	Test accuracy	Test loss
0.56	71.88	0.52	60.71	0.65

Malignant classified as Benign	46

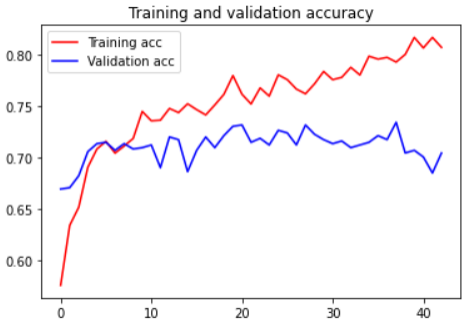


Figure 48: Accuracy

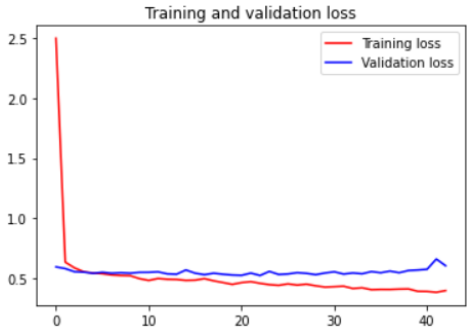


Figure 49: Loss

Smaller FC layer

The fully connected layer with 512 layers is replaced with a layer with 256 neurons.

f2 score	Validation accuracy	Validation loss	Test accuracy	Test loss
0.55	73.96	0.52	61.90	0.69

Malignant classified as Benign	48

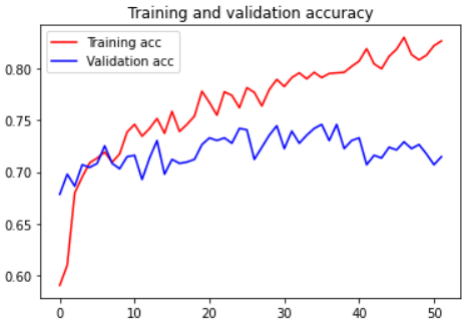


Figure 50: Accuracy

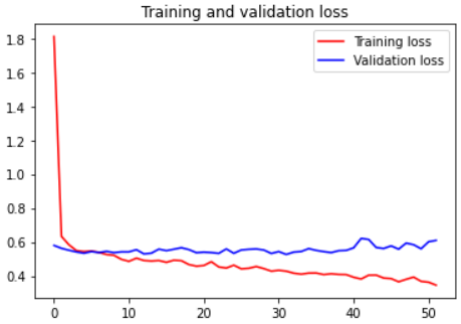


Figure 51: Loss

The network learns very fast to distinguish between benign and malignant, with about 78% accuracy on training set in the best model. On the other hand, by looking at the loss graph, it begins overfitting very soon: some regularization method is very needed.

With dropout

Starting from the vanilla model (which has the higher f2 score), dropout is added at the end of the network to alleviate overfitting.

f2 score	Validation accuracy	Validation loss	Test accuracy	Test loss
0.54	71.61	0.51	62.50	0.61

Malignant classified as Benign	51

As expected, dropout mitigated the effect of overfitting, even if the f2 score has decreased.

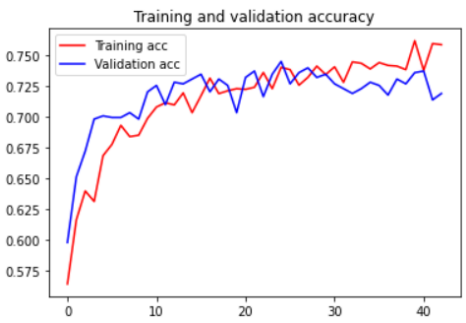


Figure 52: Accuracy

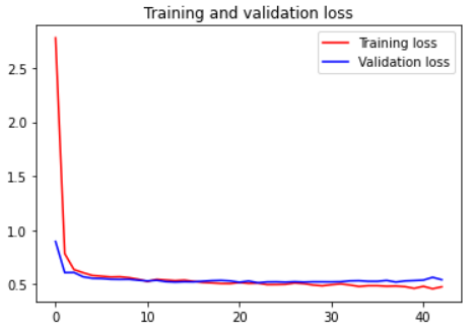


Figure 53: Loss

Two FC layers

To try to improve performance, an additional fully connected layer is added so that there is one with 512 neurons and one with 256 plus the last layer with one sigmoidal neuron.

f2 score	Validation accuracy	Validation loss	Test accuracy	Test loss
0.57	73.18	0.52	63.69	0.68

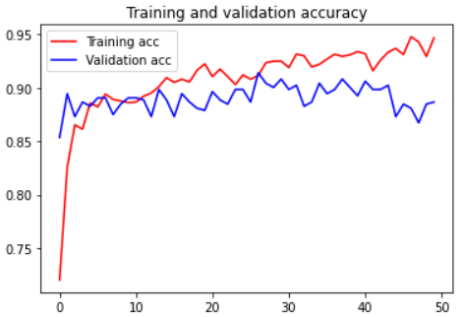


Figure 54: Accuracy

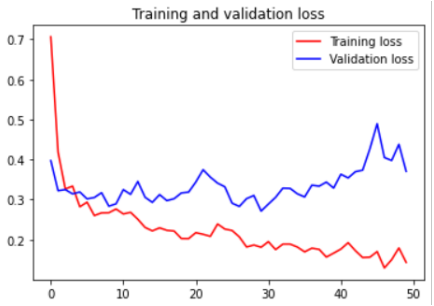


Figure 55: Loss

6.6.2 Fine tuning

One layer

Starting from the model 4, the 3rd Conv2D layer in the 5th block is unfrozen and the network trained.

f2 score	Validation accuracy	Validation loss	Test accuracy	Test loss
0.59	76.56	0.49	63.10	0.68

Malignant classified as Benign	43

Two layers

The same process can be applied iteratively starting from the 1-layer fine tuned model and unfreezing the last two convolutional layers.

f2 score	Validation accuracy	Validation loss	Test accuracy	Test loss
0.63	74.87	0.49	70.83	0.57

Malignant classified as Benign	41

Thanks to fine tuning, the NN scored 70.83% accuracy on the testing set, with a gain of 10% compared to the first feature extraction VGG16 model also increasing the f2 score.

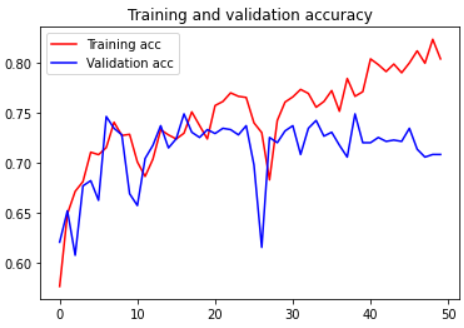


Figure 56: Accuracy

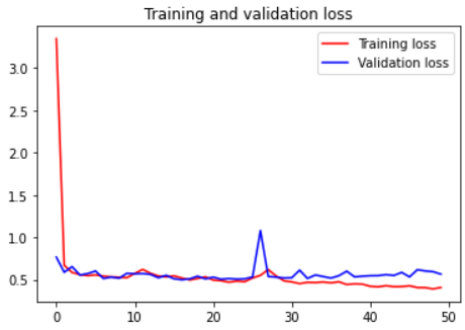


Figure 57: Loss

6.7 ResNet50 benign and malignant

6.7.1 Features extraction

Vanilla

A plain ResNet50 with two final FC layers, the former with 512 neurons and the latter with a sigmoidal neuron.

f2 score	Validation accuracy	Validation loss	Test accuracy	Test loss
0.62	73.44	0.50	66.67	0.62

Malignant classified as Benign	40

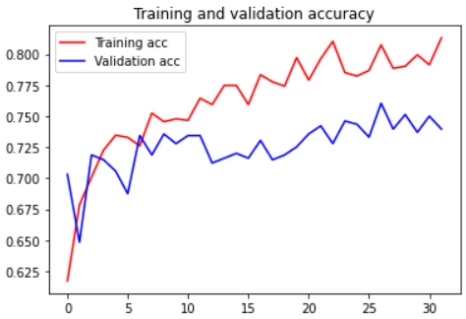


Figure 58: Accuracy

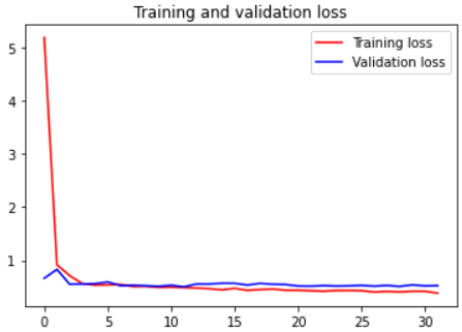


Figure 59: Loss

Smaller FC layer

The fully connected layer with 512 layers is replaced with a layer with 256 neurons.

f2 score	Validation accuracy	Validation loss	Test accuracy	Test loss
0.61	74.87	0.48	69.34	0.61

Malignant classified as Benign	43

With dropout
Starting from the vanilla model (which has the higher f2 score), dropout is added at the end of the network to alleviate overfitting.

f2 score	Validation accuracy	Validation loss	Test accuracy	Test loss
0.71	76.95	0.47	67.56	0.63

Malignant classified as Benign	25

As expected, dropout mitigated the effect of overfitting, but it was not enough to eliminate it.

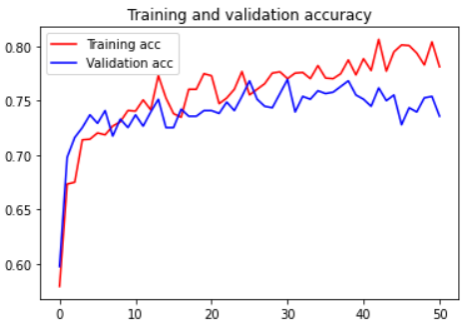


Figure 60: Accuracy

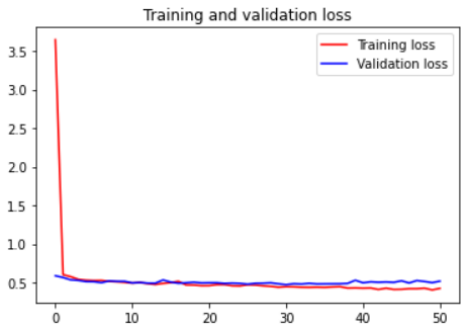


Figure 61: Loss

Two FC layers
The fully connected layers become 2 one with 512 neurons and the other with 256.

f2 score	Validation accuracy	Validation loss	Test accuracy	Test loss
0.60	74.35	0.48	67.85	0.59

Malignant classified as Benign	44

6.7.2 Fine tuning

Two layers
Starting from the model 3, the last 2 convolutional layers is unfrozen and the network trained.

f2 score	Validation accuracy	Validation loss	Test accuracy	Test loss
0.66	77.47	0.46	69.94	0.60

Malignant classified as Benign	35

6.8 InceptionV3 benign and malignant

Vanilla
A plain InceptionV3 with two final FC layers, the former with 512 neurons and the latter with a sigmoidal neuron.

f2 score	Validation accuracy	Validation loss	Test accuracy	Test loss
0.63	72.27	0.52	66.37	0.58

Malignant classified as Benign	39

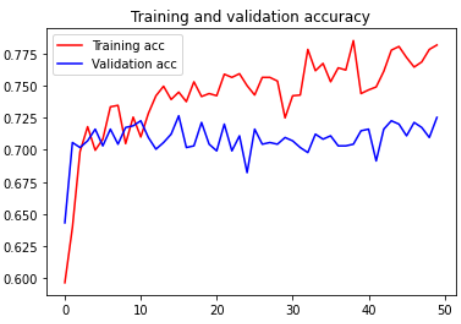


Figure 62: Accuracy

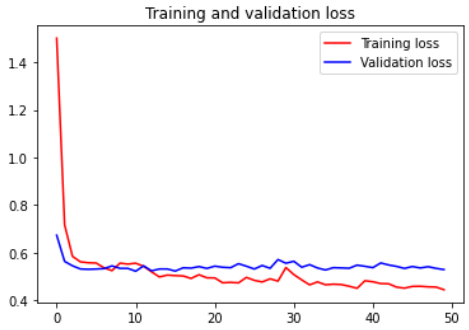


Figure 63: Loss

Smaller FC layer
The fully connected layer with 512 layers is replaced with a layer with 256 neurons.

f2 score	Validation accuracy	Validation loss	Test accuracy	Test loss
0.62	71.48	0.51	68.15	0.59

Malignant classified as Benign	41

With dropout
Staring from the vanilla model, a dropout layer is added in order to mitigate the model overfitting.

f2 score	Validation accuracy	Validation loss	Test accuracy	Test loss
0.67	72.53	0.50	64.58	0.59

Malignant classified as Benign	31

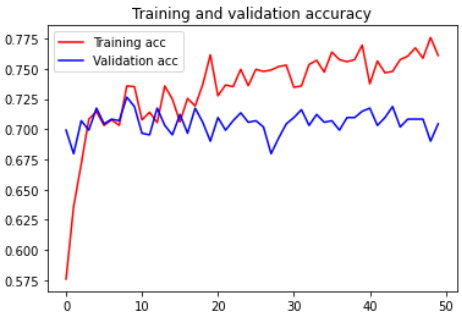


Figure 64: Accuracy

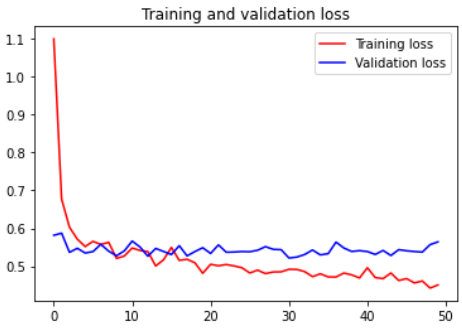


Figure 65: Loss

Two FC layers with dropout
To try to increase performance, an additional 256 FC layer was added while leaving the dropout.

f2 score	Validation accuracy	Validation loss	Test accuracy	Test loss
0.65	70.96	0.52	70.24	0.58

Malignant classified as Benign	37

6.8.1 Fine tuning

Two layers
Starting from the model 4, the last 2 convolutional layers is unfrozen and the network trained.

f2 score	Validation accuracy	Validation loss	Test accuracy	Test loss
0.64	73.18	0.50	68.45	0.58

Malignant classified as Benign	37

6.9 Error analysis

Several experiments were carried out for each pre-trained network to which only some FC and Dropout layers were added. The best result is obtained from the 3rd experiment performed with the pre-trained network ResNet50 by adding a dropout layer with a rate equal to 0.5 and a fully connected layer with 512 neurons respectively achieving the higher f2 score equals to 0.71, an accuracy of 67.56% on the test set and an AUC of 0.76. In this task the main interest is in misclassifications of malignant abnormalities as it is much more serious to get this type of class wrong.

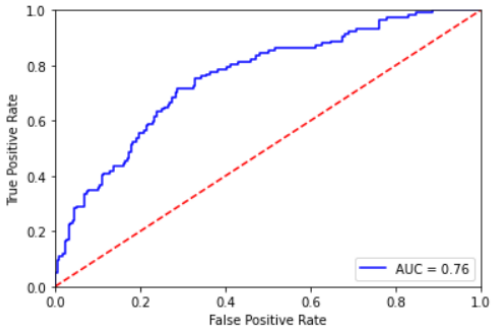


Figure 66: AUC

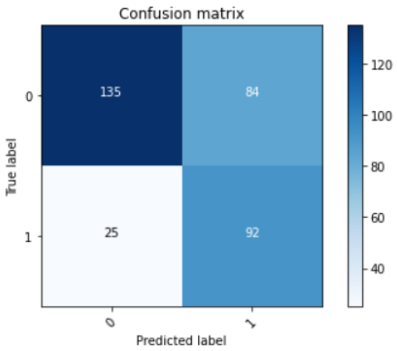


Figure 67: Confusion matrix

As can be seen from the confusion matrix, the model misses 109 out of 336 images, but only 25 malignant abnormality classified as benign:

Number of test images	336
Mispredictions	109
Malignant classified ad Be	25

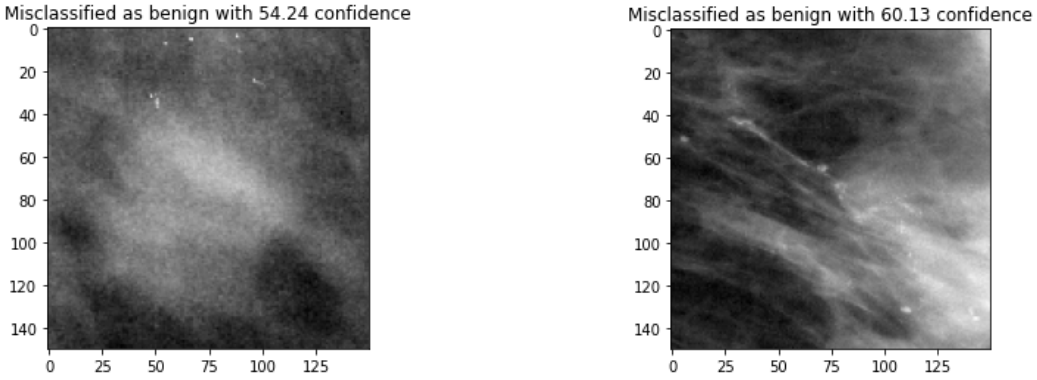


Figure 68: Example of misclassification images

7 Task 4: CNN using the Baseline

A baseline patch is patch of healthy tissue adjacent to the abnormality patch that has been extracted from the original image and provided in the dataset with the label '0'.
These images do not show any evidence of disease, however it can be assumed that they still contain information about the neighboring anomaly. Even if this assumption is found to be false, i.e., no hint of mass/calcification is present, such baseline images can still be exploited to normalize the anomaly patches by subtracting irrelevant features.
The goal of this task is to evaluate the influence of providing the model with the baseline patches, other than just the abnormality patches, for the classification of abnormality type or diagnosis.

7.1 Two classes: mass and calcification

7.1.1 Siamese Network

It is a class of neural network architectures that contain two or more identical subnetworks, meaning that they have the same configuration with the same parameters and weights and parameter updating is mirrored across both sub-networks. It is used to find the similarity of the inputs by comparing its feature vectors. More precisely, this network combine two inputs using two identical sub-networks and connecting the output of this two subtracting the output of them and taking the absolute value of the result.

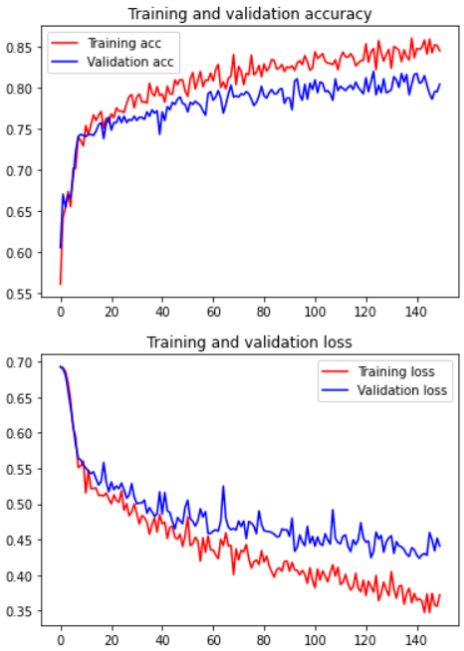
In this task, the purpose of the Siamese network is to calculate the difference between the baseline patch images and the abnormality images and use it to predict the type of abnormality. More precisely, by subtracting the features of the input images, the aim is to isolate and enhance those features that originate from masses and calcifications, attempting to improve the reliability of the classification.

Dataset creation: As done in the task 2.1, label '1' and '2', which both correspond to masses, have been aggregated and label '3' and '4', which are both calcifications have been aggregated too. In this case baseline patch images have not been discarded but concatenated with the abnormality images in the second dimension. Then, the all the images have been transformed into a float32 array

and normalized between 0 and 1 and a training and a validation set have been created.

Data Augmentation: As done in the task 2.1, data augmentation has also been performed with the same geometric transformations.

Building the model: In order to feed the input into the same convolutional base, two input channels have been created as ”left input” and ”right input” and then the two side of the siamese network have been created as ”left model” and ”right model”. Then, a Lambda layer form Keras library has been used to calculate the difference between the features of the two images and an output layer has been added to transform it into the abnormality type prediction.
The model finally achieved an accuracy on test set of 77%, which is not a great result considering that a Scratch CNN overcomes easily 80% accuracy.



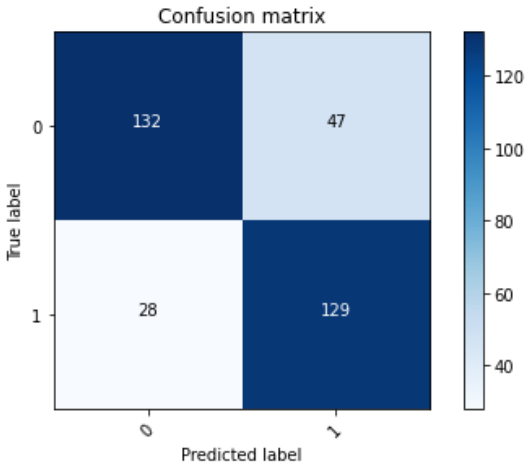


Figure 69: Training history Siamese

7.1.2 Convolutional 2D

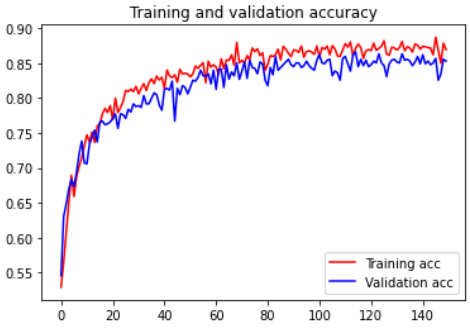
The idea of this task is to try to overlap the baseline patch images and abnormality images as if they were two channels of the same image. It is similar to the previous task but they feed a convolutional neural network.

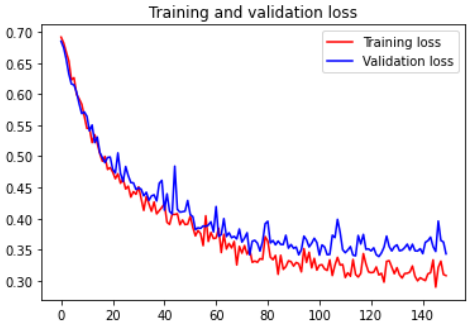
Dataset creation: As done in the previous task, the baseline patch images have not been discarded but overlapped with the abnormality images in the second dimension and aggregated according to the abnormality type. Then, the all the images have been transformed into a float32 array and normalized between 0 and 1 and a training and a validation set have been created.

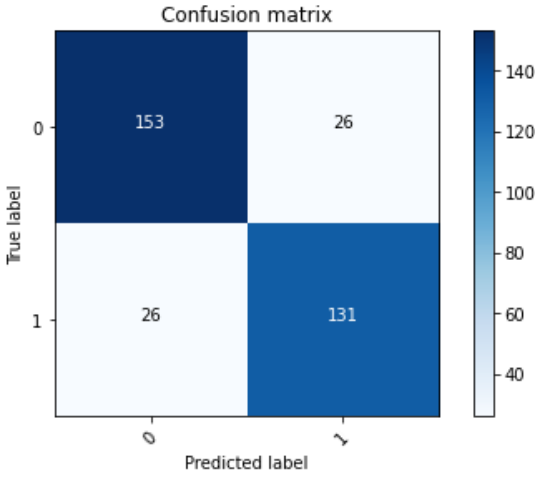
Data Augmentation: As done in the task 2.1, data augmentation has also been performed with the same geometric transformations.

Building the model: Some experiments have been carried out in order to try different configurations of convolutional neural networks. The best one was the model described below, with an accuracy on test set of 84% and the loss and accuracy history that seems smooth and good as expecting.
Anyway due to the fact that the best model of the task 3.1 achieves an accuracy of 91% using a pretrained network, it is possible to conclude that with this model the baseline patche don't improve the classification performances.

```
model7 = models.Sequential()
model7.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))
model7.add(layers.MaxPooling2D((2, 2)))
model7.add(layers.Conv2D(64, (3, 3), activation='relu'))
model7.add(layers.MaxPooling2D((2, 2)))
model7.add(layers.Conv2D(64, (3, 3), activation='relu'))
model7.add(layers.MaxPooling2D((2, 2)))
model7.add(layers.Conv2D(128, (3, 3), activation='relu'))
model7.add(layers.MaxPooling2D((2, 2)))
model7.add(layers.Flatten())
model7.add(layers.Dense(128, activation='relu'))
model7.add(layers.Dense(64, activation='relu'))
model7.add(layers.Dense(1, activation='sigmoid'))
)
```







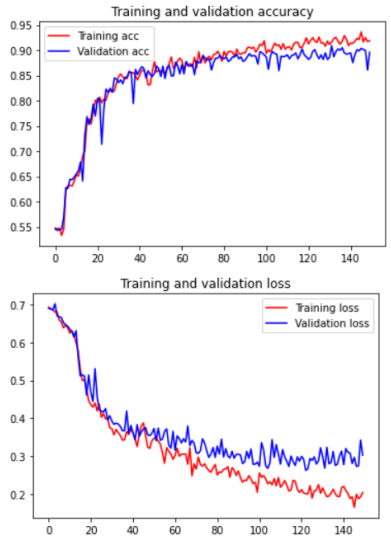
7.1.3 Convolutional 3D

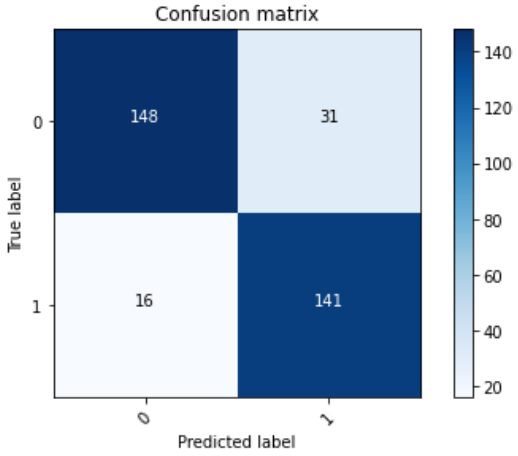
Actually, it would not be correct to consider the images as two channels of the same image because it prevents the network from learning autonomously how to combine them, so the convolutional 3D model was tried: each image is convolved with a series of filters and each of these filters returns a new feature map with depth 2, so that the images are not mixed during convolution. The results were good, in some tests exceeding the simple CNN scratch:

Data Augmentation: As done in the task 2.1, data augmentation has also been performed with the same geometric transformations.

Building the model: One test worth mentioning is a change in the way we combine feature maps: in the 3D convolution above each feature map is split and then flattened together as NN input, in a test that was done we use a 3D filter with the same depth as the input (two, like the images) and then flatten, so the feature maps were combined with the convolution. The accuracy on the test set is 86%, the loss and history of accuracy seems smooth and good as expected, this seems a good

way to combine images, several settings and small changes were tried to improve further, but nothing major.





7.1.4 Error analysis

The last chapter summarises the errors made by the CNN best model, i.e. Conv 3D in classifying the images in the test set.

Number of test images	336
Mispredictions	47

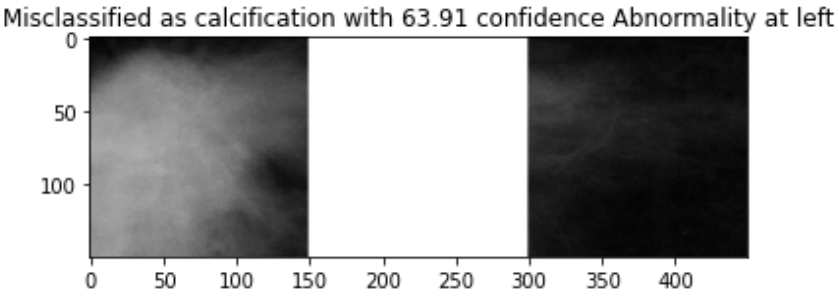
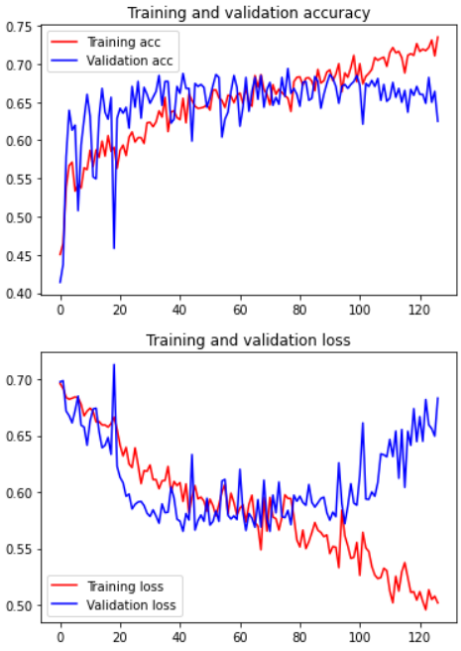
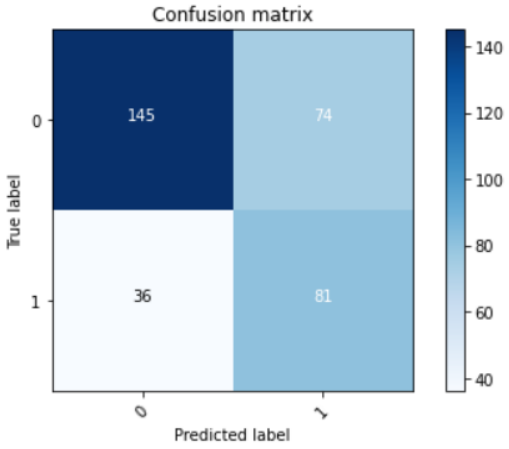


Figure 70: Example of misclassified images

7.2 Two classes: benign and malignant

A similar group of CNNs was tested on this task, the only one that has comparable results to the CNN scratch was the same one that performs well on the previous task, i.e. the first Convolutional 3D, model consisting of 5 convolutional layers alternating with MaxPooling layers and finally 4 dense layers.



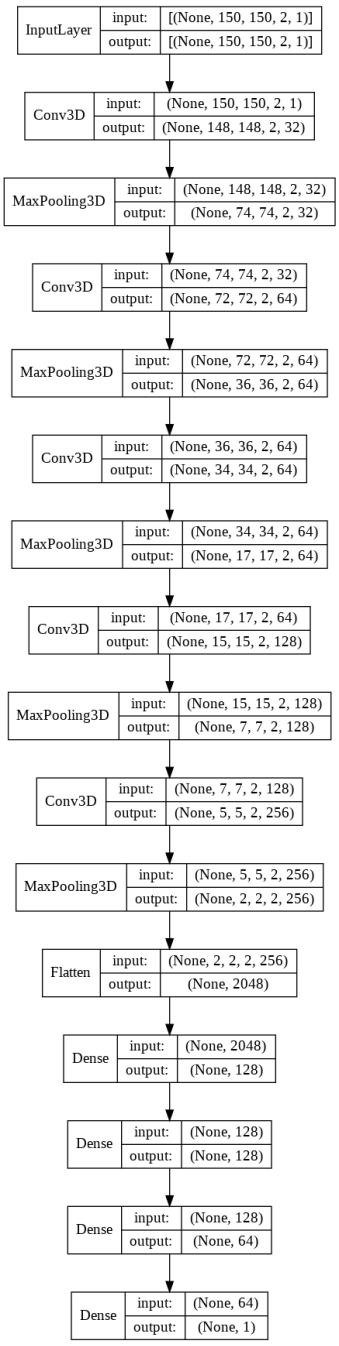


7.2.1 Error analysis

Let's take a look at the results obtained from the Convolutional 3D model.

Number of test images	336
Mispredictions	110
Malignant classified as Benign	36

Let’s take a look at the architecture of the Convolutional 3D model:



8 Task 5: CNN using Ensemble

Generally, neural networks have a high variance and it can be frustrating when trying to develop a final model to use for making predictions. A successful approach to reducing the variance of neural network models is to train multiple models instead of a single model and to combine the predictions from these models. This is called ensemble learning and not only reduces the variance of predictions but also can result in predictions that are better than any single model.

Model averaging is an ensemble learning technique where multiple sub-models contribute equally to a combined prediction and that reduces the variance in a final neural network model. Once the models have been trained, each one can be used to make a prediction and the predictions can be combined using some classifier.
This can be extended further by training an entirely new model to learn how to best combine the contributions from each submodel. This approach is called stacked generalization and can result in better predictive performance than any single contributing model.

Ensemble creation: The ensemble model was built performing the following steps:

1. Loading of the trained models
2. Aggregation of the models to create the stacked model
3. Collecting the predictions from each single trained model
4. Fitting the stacked model with a classifier or a neural network
5. Predicting the outcome

8.1 Two classes: mass and calcification

The aim of this task is to develop a composite classifier (Ensemble of Neural Network) to boost classification performance for discriminating the abnormality type, in this case between mass and calcification.

Dataset creation: As done in the task 2.1, from the original dataset with five labels, the label '0', which correspond to baseline patch, has been discarded because this task considers only the abnormalities, and label '1' and '2', which both correspond to masses, have been aggregated and label '3' and '4', which are both calcifications have been aggregated too. Then, the images have been transformed into a float32 array and normalized between 0 and 1 and a training and a validation set have been created.

8.1.1 Using from scratch models

Some experiments have been carried out using the model 9 and 10 trained in task 2.1 to create the ensemble and then their predictions have been combined using a classifier or others layers of neural networks.

Some experiments: Logistic Regression, Linear SVC and SVC The first experiment combines the predictions of the two models through a logistic regression model. The second experiment instead uses a linear support vector machine classifier and the third a simple support vector machine classifier. All the experiments have achieved an accuracy on the test set of 87% but with confusion matrices a little bit different.

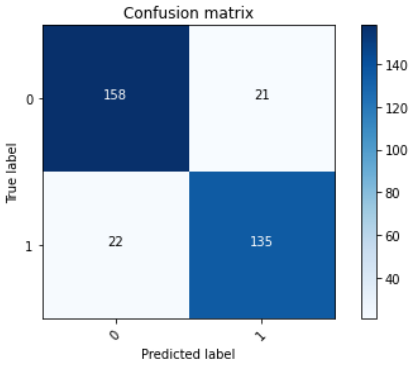


Figure 71: Logistic Regression

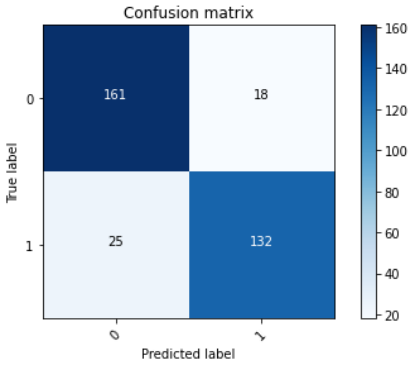


Figure 72: Linear SVC

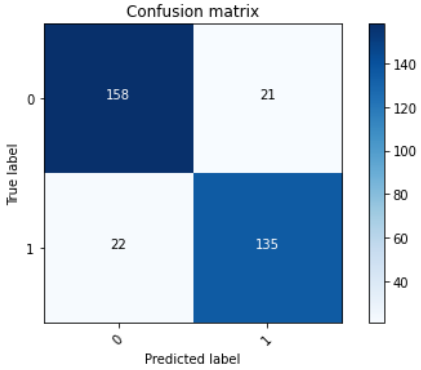


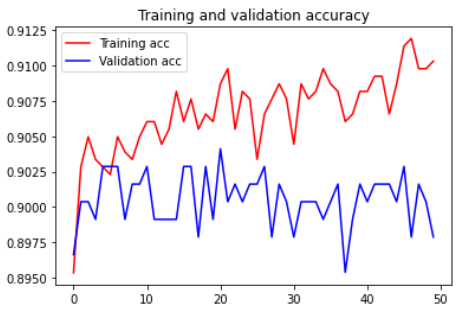
Figure 73: SVC

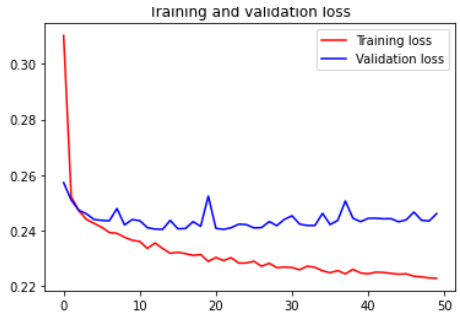
8.1.2 Using intermediate layers of models from scratch

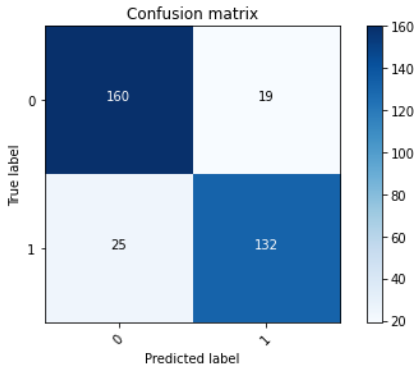
To build the ensemble model the output layer has been deleted from the models, performing the prediction directly with another output layer, which includes both models, or with classifiers used before.

```
1 intermediate_layer_model1 = keras.Model(inputs=model1.input, outputs=model1.
      get_layer("dense_1").output)
2 intermediate_layer_model2 = keras.Model(inputs=model2.input, outputs=model2.
      get_layer("dense").output)
```

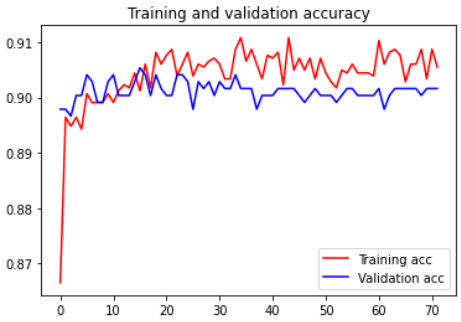
Adding an output layer The first experiment uses an output layer with a sigmoid function to make the predictions of the ensemble model, achieving an accuracy on the test set of 86,9%.

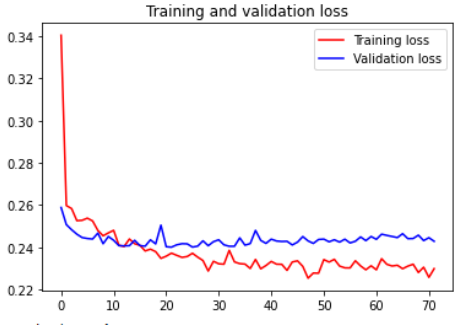


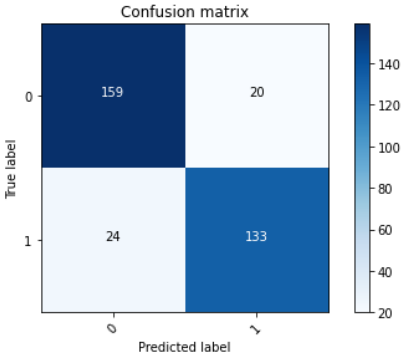




The second experiment tries to reduce overfitting adding a Dropout layer. Although the accuracy has remained the same, the network trend is more regular.







Logistic Regression, Linear SVC and SVC As done previously, some classifiers have been used to combine the output of the models deprived of the last layer. The Logistic Regression has achieved an accuracy on the test set of 87,5%, the Linear SVC of 87,2% and the SVC of 87,8%.

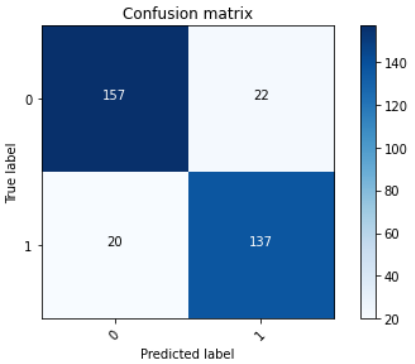


Figure 74: Logistic Regression

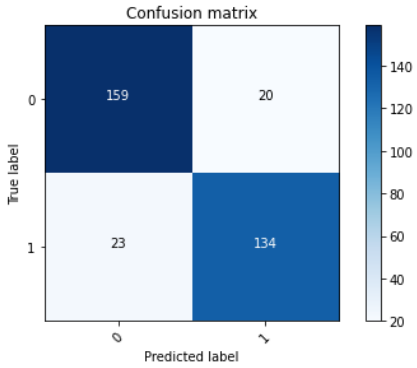


Figure 75: Linear SVC

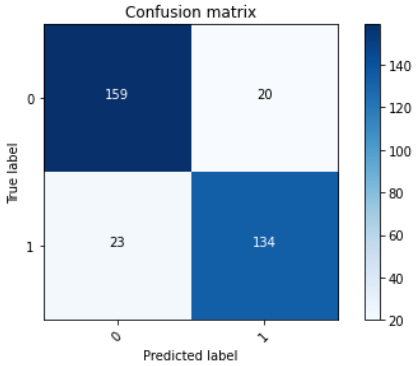


Figure 76: SVC

8.1.3 Using from scratch models and pretrained networks

The best pretrained model of task 3.1, inception 3, has been added to model 9 and 10 from task 2.1 to create the ensemble and then their predictions have been combined, as done previously, using a classifier or others layers of neural networks. Of course the needed preprocessing phase for the inception network has been introduced too.

Logistic Regression, Linear SVC and SVC The first experiment combines the predictions of the three models through a logistic regression model with an accuracy on the test set of 89,9%. The second experiment instead uses a linear support vector machine classifier,with an accuracy on the test set of 90,2%, and the third

a simple support vector machine classifier, with an accuracy on the test set of 89,9%.

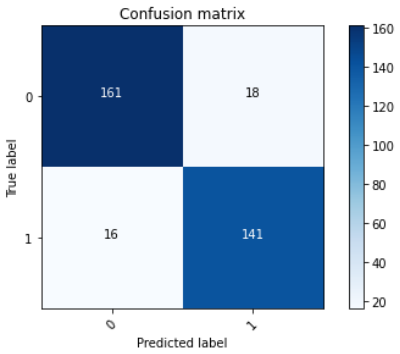


Figure 77: Logistic Regression

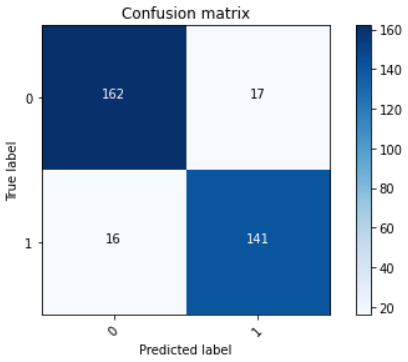


Figure 78: Linear SVC

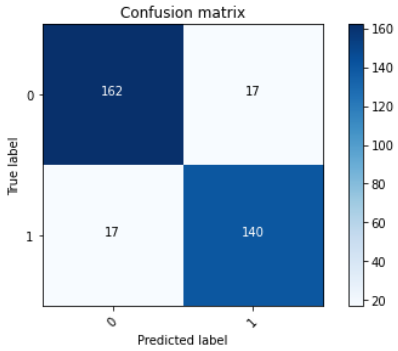


Figure 79: SVC

8.1.4 Using intermediate layers of models from scratch and pretrained networks

Again, the ensemble model has been built deleting the output layer from the models and performing the prediction directly with another output layer, which includes the three models, or with classifiers used before.

Logistic Regression, Linear SVC and SVC The first experiment combines the predictions of the three models through a logistic regression model with an accuracy on the test set of 89,3%. The second experiment instead uses a linear support

vector machine classifier,with an accuracy on the test set of 86,9%, and the third a simple support vector machine classifier, with an accuracy on the test set of 90,2%.

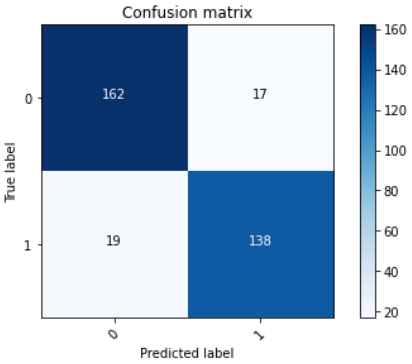


Figure 80: Logistic Regression

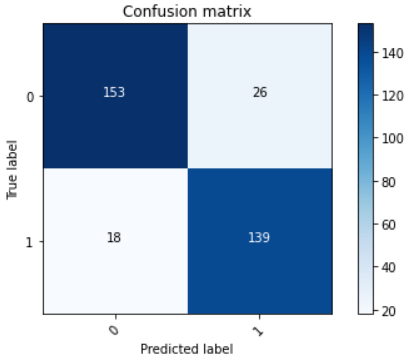


Figure 81: Linear SVC

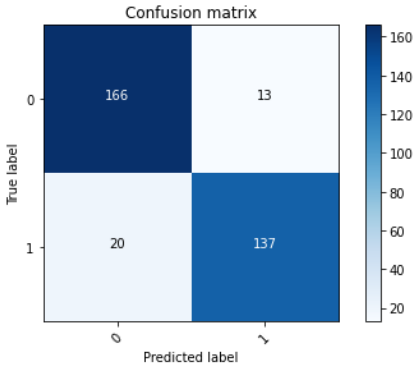
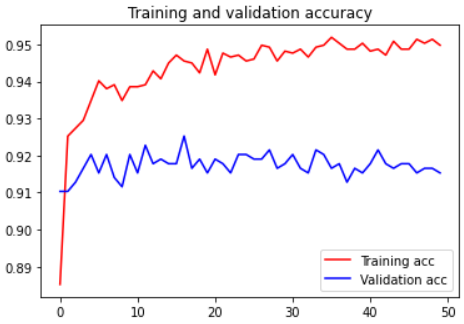
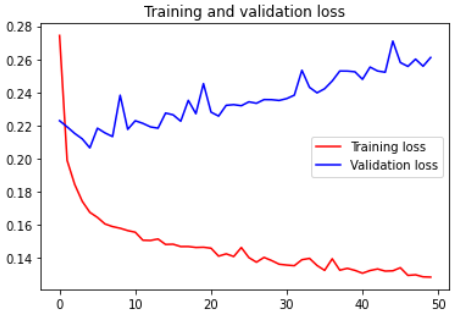
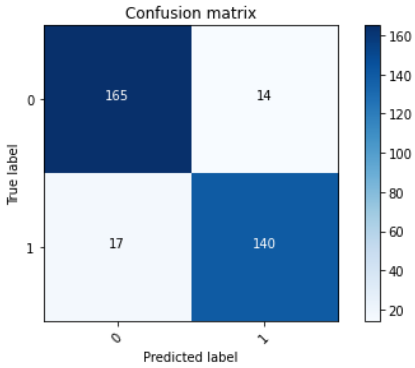


Figure 82: SVC

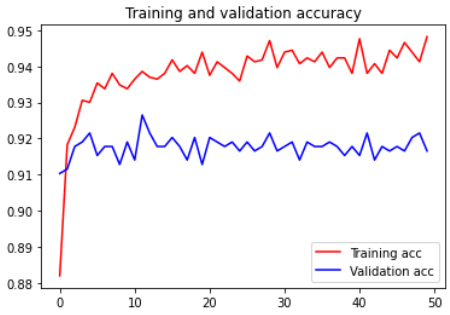
Some experiments and the best model: As done previously, the ensemble model has been built deleting the last layer of the models and merging them with an unique output layer achieving an accuracy of 90,8% on the test set.

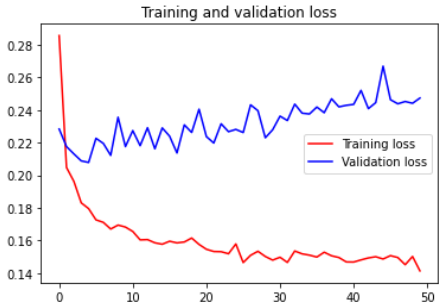


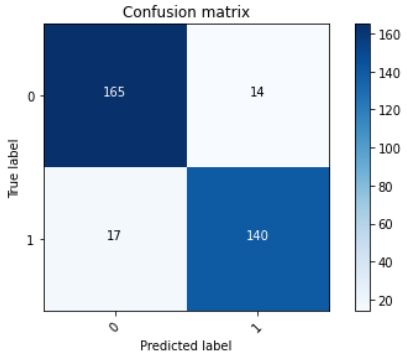




Then, it has been tried to add a dropout layer to reduce the overfitting, which slightly improved the trend of the model while maintaining the same accuracy value, so this turned out to be the best model.







8.1.5 Error analysis:

The last chapter summarises the errors made by the CNN best model, i.e. ensemble model composed by the model 10, model 9 and the pretrained inception 3 with intermediate layers and the output layer with dropout, in classifying the images in the test set.

Number of test images	336
Mispredictions	31

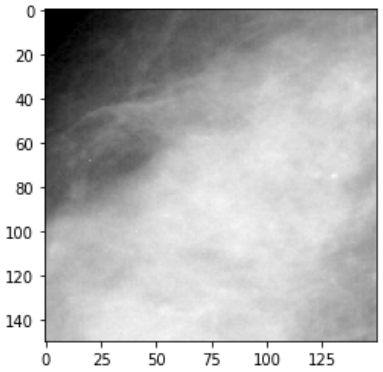


Figure 83: Misclassified as mass with 52 confidence

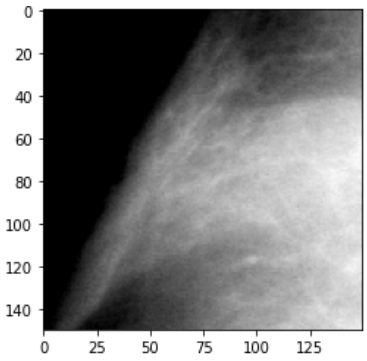


Figure 84: Misclassified as calcification with 91 confidence

8.2 Two classes: benign and malignant

Dataset creation: As done in the task 2.2, from the original dataset with five labels, the label '0', which correspond to baseline patch, has been discarded because this task considers only the abnormalities, and label '1' and '3', which both correspond to benign, have been aggregated and label '2' and '4', which are both malignant have been aggregated too. Then, the images have been transformed into a float32 array and normalized between 0 and 1 and then a training and a validation set have been created.

8.2.1 Using from scratch models

Some experiments have been carried out using the model 7 and 6 trained in task 2.2 to create the ensemble and then their predictions have been combined using a classifier or others layers of neural networks.

Some experiments: Logistic Regression, Linear SVC and SVC The first experiment combines the predictions of the two models through a logistic regression model achieving a F2 score value of 63% and an AUC of 0.70. The second experiment instead uses a linear support vector machine classifier, with a F2 score value of 57% and an AUC of 0.71, and the third a simple support vector machine classifier, with a F2 score value of 59% and an AUC of 0.70.

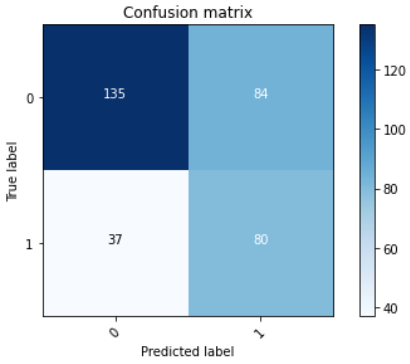


Figure 85: Logistic Regression

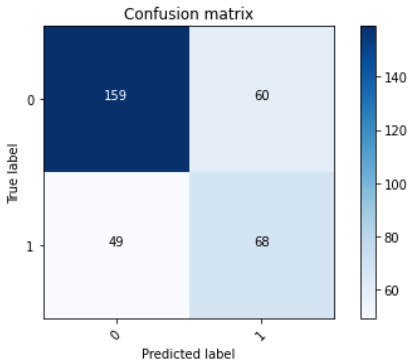


Figure 86: Linear SVC

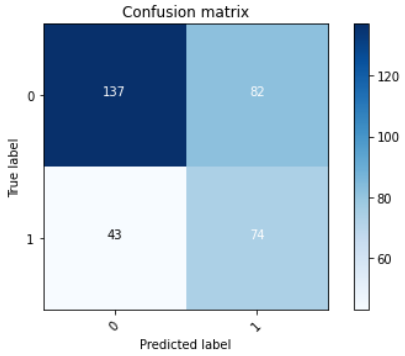


Figure 87: SVC

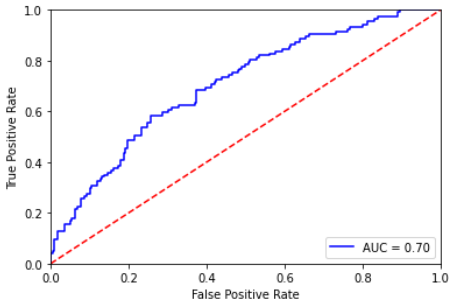


Figure 88: Logistic Regression

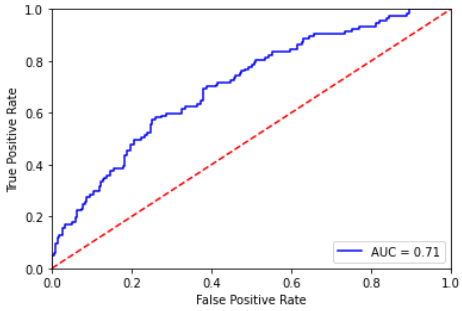


Figure 89: Linear SVC

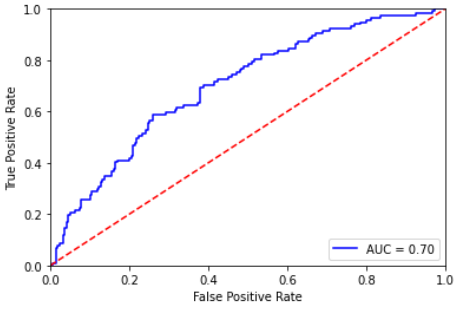


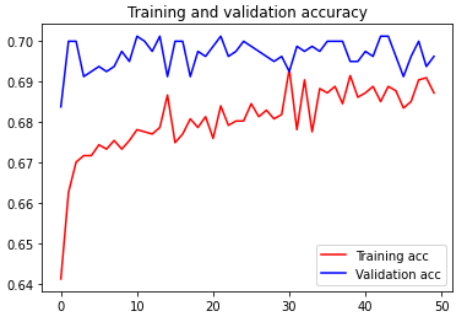
Figure 90: SVC

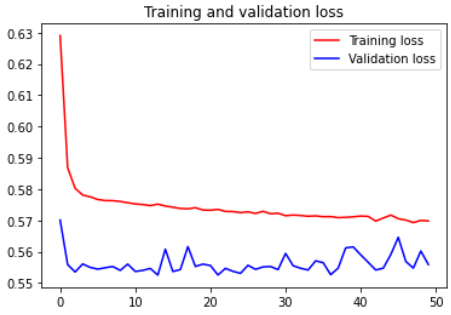
8.2.2 Using intermediate layers of models from scratch

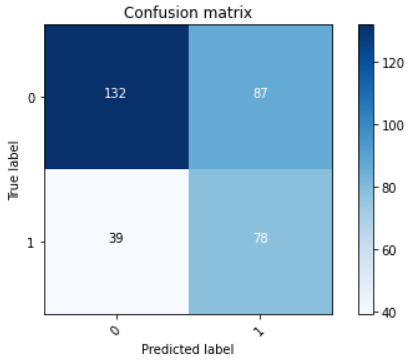
To build the ensemble model the output layer has been deleted from the models, performing the prediction directly with another output layer, which includes both models, or with classifiers used before.

```
intermediate_layer_model1 = keras.Model(inputs=model1.input, outputs=model1.
    get_layer("dense_1").output)
intermediate_layer_model2 = keras.Model(inputs=model2.input, outputs=model2.
    get_layer("dense").output)
```

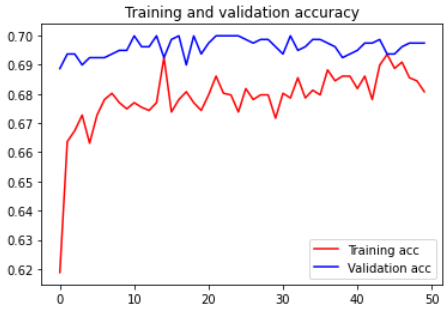
Adding an output layer The first experiment uses an output layer with a sigmoid function to make the predictions of the ensemble model, achieving a F2 score value of 61,6% and an AUC of 0.70.

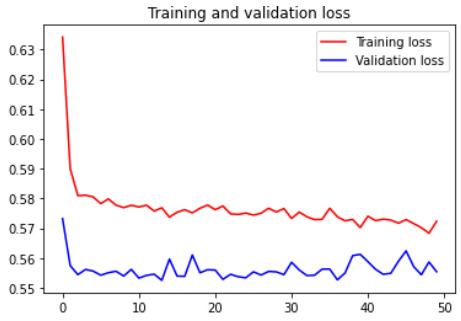


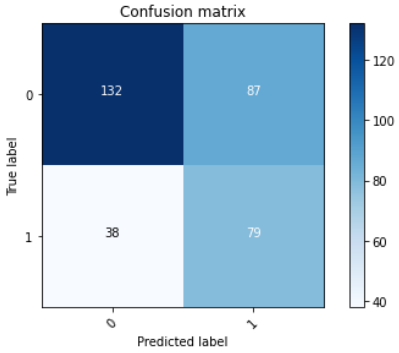




The second experiments adds a Dropout layer in order to reduce the overfitting, achieving a F2 score value of 62,3% and an AUC of 0.70.







Logistic Regression, Linear SVC and SVC As done previously, some classifiers have been used to combine the output of the models deprived of the last layer. The Logistic Regression has achieved a F2 score value of 63,2% and an AUC of 0.70, the Linear SVC of 53,3% with an AUC of 0.69, and the SVC of 56,9%.

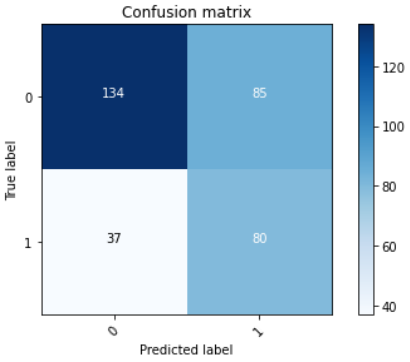


Figure 91: Logistic Regression

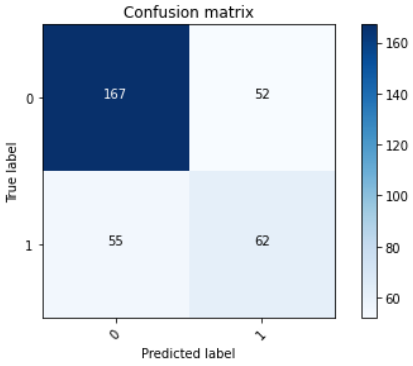


Figure 92: Linear SVC

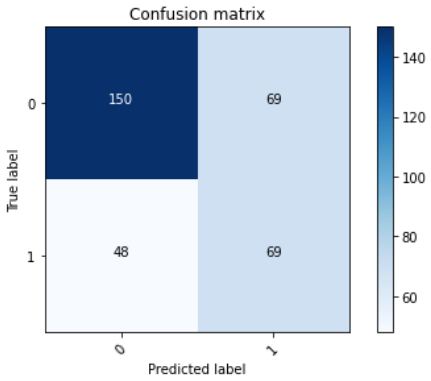


Figure 93: SVC

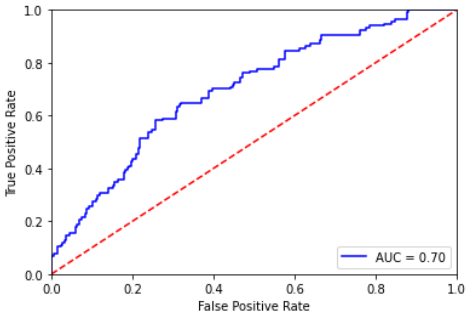


Figure 94: Logistic Regression

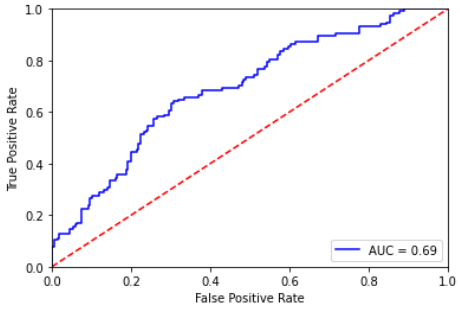


Figure 95: Linear SVC

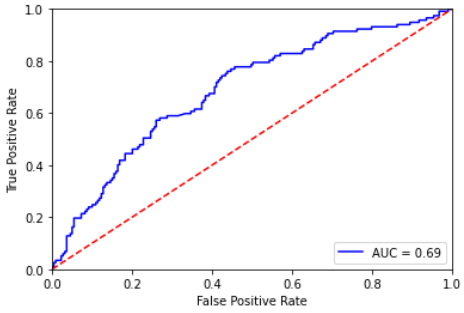


Figure 96: SVC

8.2.3 Using from scratch models and pretrained networks

The best two pretrained models of task 3.2, inception 4 and resnet 3, has been added to model 7 from task 2.2 to create the ensemble and then their predictions have been combined, as done previously, using a classifier or others layers of neural networks. Of course the needed preprocessing phase for the inception and resnet networks has been introduced too.

Logistic Regression, Linear SVC and SVC The first experiment combines the predictions of the three models through a logistic regression model with a F2 score value of 71,5% and an AUC of 0.77. The second experiment instead uses a linear SVM classifier, with a F2 score value of 67,8% and an AUC of 0.77, and the third a simple SVM classifier, with a F2 score value of 67,4% and an AUC of 0.73.

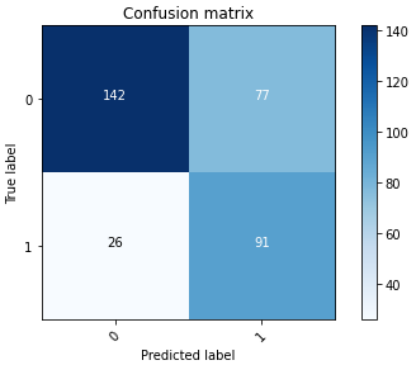


Figure 97: Logistic Regression

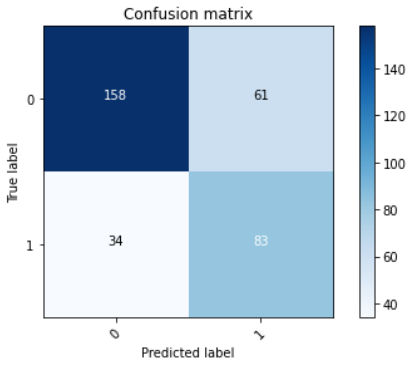


Figure 98: Linear SVC

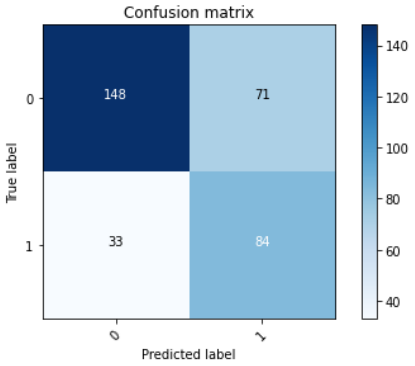


Figure 99: SVC

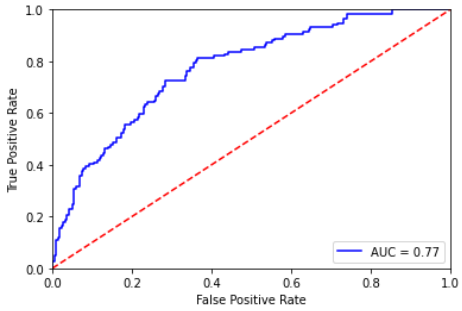


Figure 100: Logistic Regression

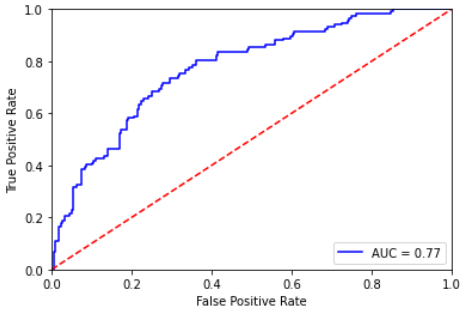


Figure 101: Linear SVC

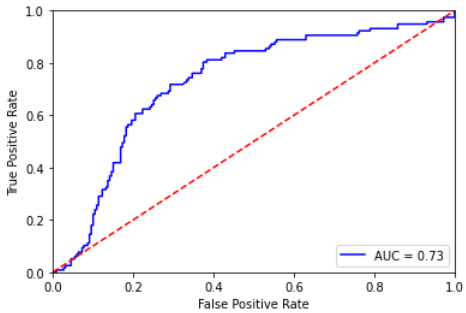


Figure 102: SVC

8.2.4 Using intermediate layers of models from scratch and pretrained networks

Again, the ensemble model has been built deleting the output layer from the models and performing the prediction directly with another output layer, which includes the three models, or with classifiers used before.

Logistic Regression, Linear SVC and SVC The first experiment combines the predictions of the three models through a logistic regression model with a F2 score value of 69,8% and an AUC of 0.75. The second experiment instead uses a linear support vector machine classifier,with a F2 score value of 65% and an AUC of 0.76, and the third a simple support vector machine classifier, with a F2 score value of 61,5% and an AUC of 0.75.

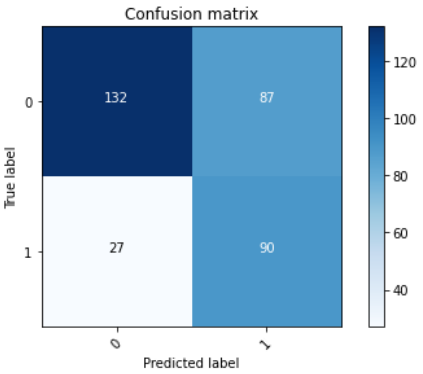


Figure 103: Logistic Regression

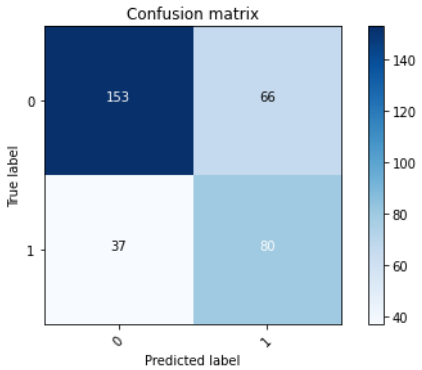


Figure 104: Linear SVC

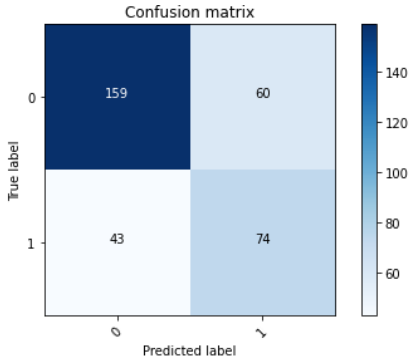


Figure 105: SVC

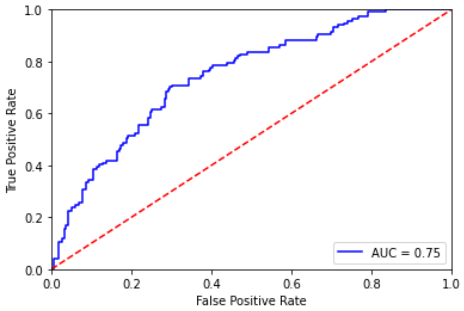


Figure 106: Logistic Regression

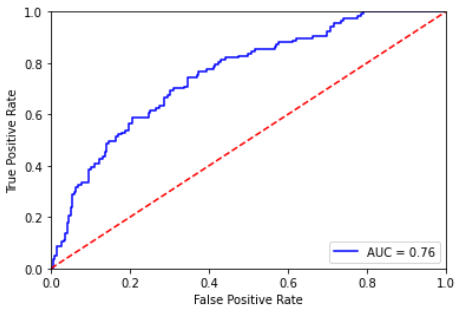


Figure 107: Linear SVC

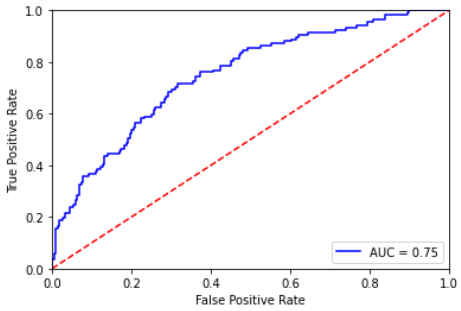
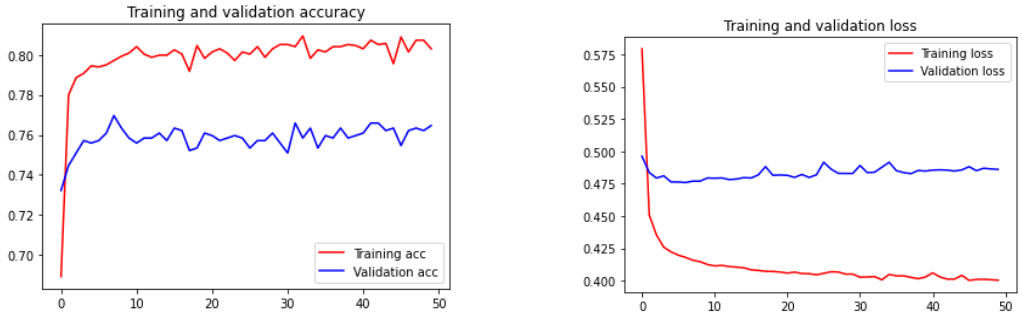
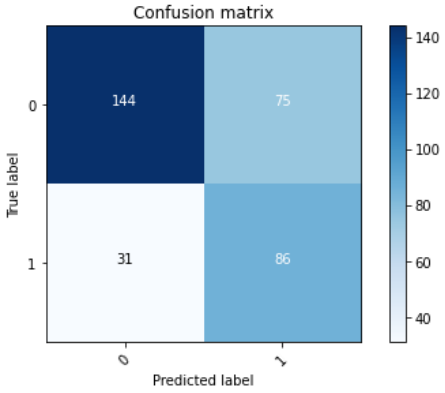


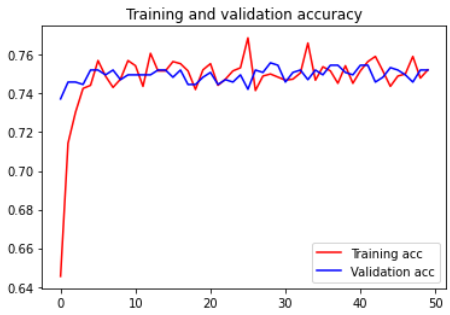
Figure 108: SVC

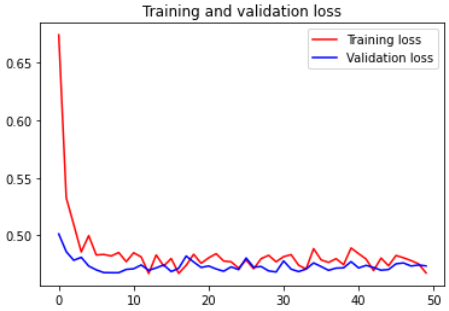
Adding an output layer: As done previously, the ensemble model has been built deleting the last layer of the models and merging them with an unique output layer. The result was a F2 score value of 68,4% and an AUC of 0.75

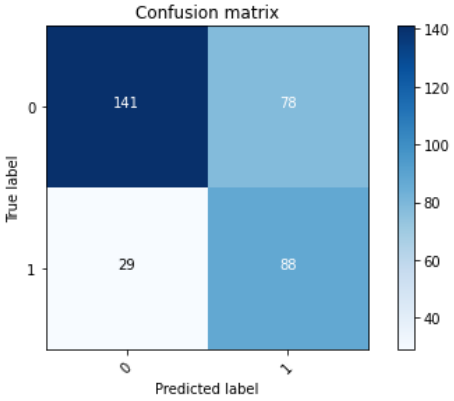




Then, it has been tried to add a dropout layer to reduce overfitting. The result achieved was a F2 score value of 69,4% and an AUC of 0.77.







Only pretrained: It has been tried to build the ensemble model only with the pretrained inception 4 and resnet 3, deleting the last layer and combining their prediction with Logistic Regression. The result achieved by the ensembled model was an f2 score value of 69,8% on the test set and an AUC of 0.75.

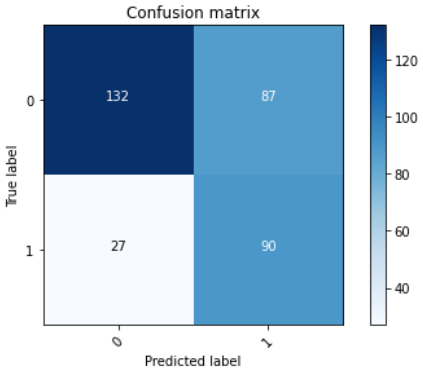


Figure 109: Confusion matrix

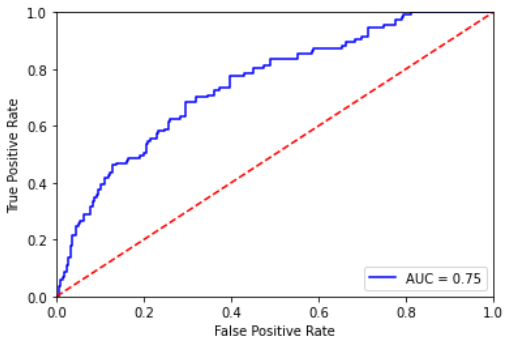


Figure 110: AUC

Best model and error analysis: The best model turned out to be the ensemble built with the model 7 from task 2.2 and the pretrained networks inception 4 and resnet 3 combined with a logistic regression, without deleting their output layers. Due to the fact that the F2 score measure has been used to assess the goodness of a classifier, the error analysis has been performed considering only the misclassification of malignant cases classified instead as benign.

Number of test images	336
Mispredictions	26

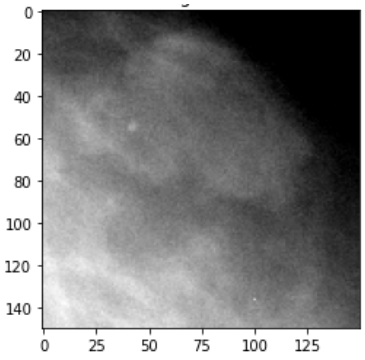


Figure 111: Misclassified as benign with 52 confidence

8.2.5 Threshold moving for imbalanced classification

So far the F2 score value and the confusion matrix have been calculated considering the default threshold of 0.5, through which all values equal or greater than the threshold are mapped to one class and all other values are mapped to another class.
To improve even more the performance of a classifier for an imbalanced classification problem, it is possible to tune this threshold used to map probabilities to class labels. The idea is to use different threshold values to calculate the F2 score metric and the threshold that achieves the best evaluation metric is then adopted for the model when making predictions.

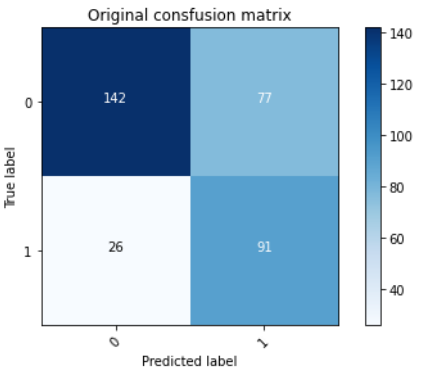
For this experiment has been decided to use the best ensemble model from task 5.2, in order to improve the classification between benign and malignant class,

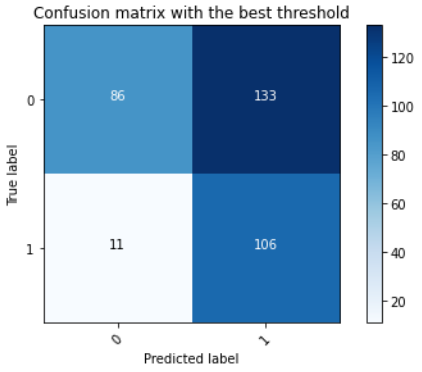
taking into consideration that a misclassification of a malignant image is more costly and dangerous.

Calculation of F2 score for each threshold As anticipated before, the idea is to compute the F2 score value for each threshold and then choose the best one for the specific purpose. In order to do so, the predictions of the ensemble model have been taken and the precision-recall curve has been considered to obtain the values of precision and recall for each threshold. Finally, different F2 score values have been computed using all the precision and recall values and the highest F2 score gives the best threshold.

```
train_pred = stacked_prediction(members, model, train_images)

# Calculate the f2 score for each threshold on the training set
precision, recall, thresholds = precision_recall_curve(train_labels, train_pred)

beta = 2
fbeta = ((1 + beta**2) * precision * recall) / (beta**2 * precision + recall)

# Locate the index of the highest f2 score
ix = np.argmax(fbeta)
best_threshold = thresholds[ix]
```

Results As expected, moving the threshold to a better value improves the classification. The best threshold turned out to be 0.21, very different from the default value of 0.5.
Infact, using this threshold, it is been able to improve the F2 score value on the test set from 71,5% to 75%, reducing the number of misclassification of malignant images from 26 to 11 and increasing the number of correct classification of malignant from 91 to 106, as it is possible to see from confusion matrices.





References

[1] This CBIS-DDSM (Curated Breast Imaging Subset of DDSM) is an updated and standardized version of the Digital Database for Screening Mammography (DDSM).
https://wiki.cancerimagingarchive.net/display/Public/CBIS-DDSM

[2] CUI Xiao-mei, YU Xiang and YANG Fan.
"Five Classification of Mammography Images Based on Deep Cooperation Convolutional Neural Network", 2019.
https://www.asrjetsjournal.org/index.php/American_Scientific_Journal/article/view/4942

[3] Li Shen, Laurie R. Margolies, Joseph H. Rothstein, Eugene Fluder, Russell McBride and Weiva Sieh.
"Deep Learning to Improve Breast Cancer Detection on Screening Mammography", 2019.
https://www.nature.com/articles/s41598-019-48995-4

[4] Ridhi Arora, Prateek Kumar Rai and Balasubramanian Raman.
"Deep feature–based automatic classification of mammograms", 2020.
https://link.springer.com/article/10.1007/s11517-020-02150-8

[5] Lenin G. Falconi, Maria Perez, Wilbert G. Aguilar, Aura Conci.
"Transfer Learning and Fine Tuning in Breast Mammogram Abnormalities Classification on CBIS-DDSM Database", 2020.
https://astesj.com/v05/i02/p20/

[6] Mei-Ling Huang, Ting-Yu Lin.
"Dataset of breast mammography images with masses", 2020.
https://www.sciencedirect.com/science/article/pii/S2352340920308222#:~:text=Through%20data%20augmentation%2C%20the%20number,classify%20these%20breast%20mammography%20images

[7] John A. Swets.
"Measuring the Accuracy of Diagnostic Systems".

http://wixtedlab.ucsd.edu/publications/Psych%20218/
Swets_1988.pdf

[8] Ayoosh Kathuria.
"Intro to Optimization in Deep Learning: Busting the Myth About Batch Normalization".
https://blog.paperspace.com/busting-the-myths-about-batch-normaliza

[9] Dmytro Mishkin.
"Batch normalization before or after Relu.".
https://github.com/ducha-aiki/caffenet-benchmark/
blob/master/batchnorm.md