MSc Artificial Intelligence and Data Engineering

Project Report

Gianluca Serao - Lorenzo Barigliano
Computational Intelligence and Deep Learning course - a.a. 2020/2021

Table of contents

Introduction

This is the report for the project of computational intelligence and deep learning course. The project aims to develop different models to solve binary classification problems on the **CBIS DDSM: Curated Breast Imaging Subset of Digital Database for Screening Mammography** dataset. In particular, we will perform two binary classification tasks, breast abnormalities classification and breast abnormalities diagnosis, using different CNNs. In particular, we will perform the tasks with CNNs trained from scratch, with pre-trained CNNs, with a siamese network and using an ensemble technique.

State-of-the-art

As the first step, we searched for several papers concerning breast abnormality detection and diagnosis (benign/malignant) using deep learning techniques to understand the state-of-the-art technologies for these tasks. Roughly all techniques are based on convolutional neural networks. The works based on CNN trained from scratch exploit image pre-processing to perform multi-scale detection or image enhancement. Makandar and Halalli used median smoothing filters and histogram equalization.[1] Lotter et al. used a multi-scale ResNet CNN to detect abnormality patches and then perform the diagnosis.[2] G. Prathibha and B. Chandra Mohan used a multiresolution transform approach on the ROI (Region of Interest) and a two-layer CNN with four fully connected layers to perform classification. [3] An innovative approach is used by Chun-ming et al., they developed a deep cooperating convolutional neural network to perform a five classes classification on mammograms. The two CNN have different weights to capture more various features from the input images. [4] Most of these works use CNNs with two or more layers, smoothing filters to obtain multi-scale images.

The works based on pre-trained CNN use the most famous CNNs. From these, the most interesting are: VGG16, ResNet50, InceptionV3. Soriano et al. used fine-tuning on the InceptionV3.[5] Chougrad et al. used fine-tuning on three pre-trained models: VGG16, RestNet50 and InceptionV3.[6] Li Shen et al. used a transfer learning approach with VGG16 and Resnet50.[7]

[1] "Breast Cancer Image Enhancement using Median Filter ... - CiteFactor." https://www.citefactor.org/journal/pdf/Breast-Cancer-Image-Enhancement-using-Median-Filter-and-CLAHE.pdf. Accessed 12 Feb. 2021.

[2] "A Multi-Scale CNN and Curriculum Learning Strategy for ...." 21 Jul. 2017, https://arxiv.org/abs/1707.06978. Accessed 12 Feb. 2021.

[3] "Mammograms Classification Using Multiresolution Transforms and ...." https://ieeexplore.ieee.org/document/8494076. Accessed 12 Feb. 2021.

[4] "Five Classification of Mammography Images Based on Deep ...." 3 Jul. 2019, https://www.asrjetsjournal.org/index.php/American_Scientific_Journal/article/view/4942. Accessed 12 Feb. 2021.

[5] "Mammogram Classification Schemes by Using Convolutional ...." 31 Dec. 2017, https://link.springer.com/chapter/10.1007/978-3-319-72727-1_6. Accessed 12 Feb. 2021.

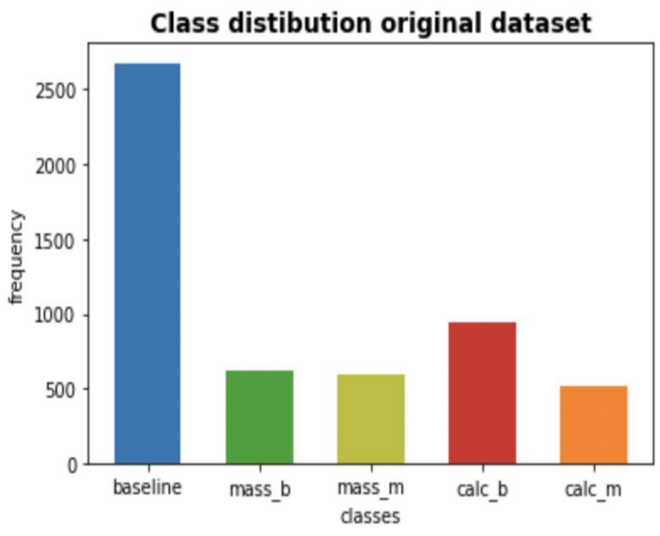
[6] "Deep Convolutional Neural Networks for breast cancer screening ...." https://www.sciencedirect.com/science/article/pii/S0169260717301451. Accessed 12 Feb. 2021.

[7] "Deep Learning to Improve Breast Cancer Detection on ... - Nature." 29 Aug. 2019, https://www.nature.com/articles/s41598-019-48995-4. Accessed 12 Feb. 2021.

Other works are based on object detection approaches. Ribli et al. used fast R-CNN for abnormality diagnosis. [8] Akselrod-Ballin et al. used faster R-CNN for abnormality detection.[9] Almost all works use data augmentation techniques, different transformation are applied to the images but the most common are rotation, flipping and shearing.
To summarize, in the literature different pre-processing techniques are used to obtain enhancements on the mammograms. Commonly, smoothing filters and contrast enhancement techniques are used. Data augmentation techniques are used for training, the most common transformations used are rotation and flipping. Different CNNs are used: the CNNs trained from scratch are at least composed of two convolutional layers, some approach instead used multiple CNNs. The pre-trained CNNs are the same used also for other tasks, such as VGG16, ResNet50, InceptionV3 and others. Some works use R-CNN to perform ROI detection and classification.

Dataset

The dataset used for this project is extracted from the CBIS-DDSM. The dataset contains five kinds of patches taken from the original dataset: baseline patches, that corresponds to healthy tissue taken from the adjacency of an abnormality, and abnormality patches of four classes (mass benign, mass malignant, calcification benign and calcification malignant). For each baseline



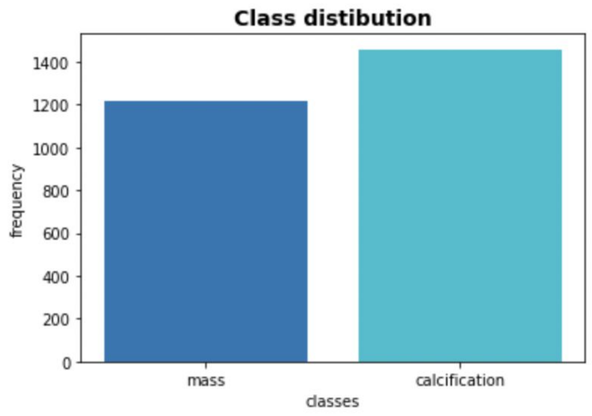
patch, we have the correspondent abnormality. So, half of the dataset is composed of baseline patches and half of the abnormality patches. All the patches are 150x150 tensors in greyscale.

To perform the binary classification tasks (abnormality classification and diagnosis), we need to binarize the dataset accordingly. In particular, for the abnormality classification, we need to

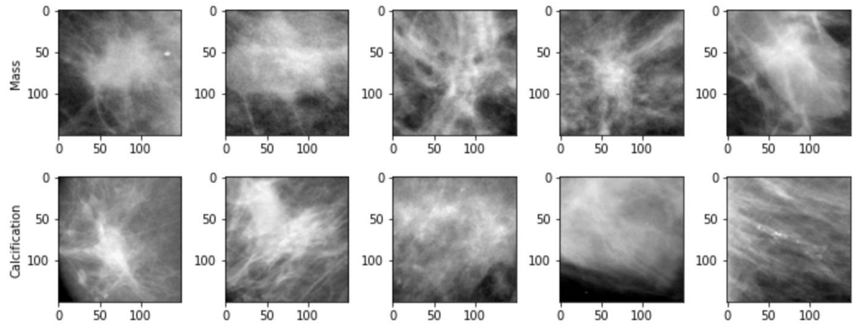
[8] "Detecting and classifying lesions in mammograms with Deep ...." 15 Mar. 2018, https://www.nature.com/articles/s41598-018-22437-z. Accessed 12 Feb. 2021.
[9] "Deep Learning for Automatic Detection of Abnormal ... - SpringerLink." 9 Sep. 2017, https://link.springer.com/chapter/10.1007/978-3-319-67558-9_37. Accessed 12 Feb. 2021.

merge mass benign and malignant into one class and calcification benign and malignant into the other. For the abnormality diagnosis instead, we need to merge benign and malignant patches belonging to mass and calcification abnormalities. The resulting class distribution after the binarization of the dataset are:

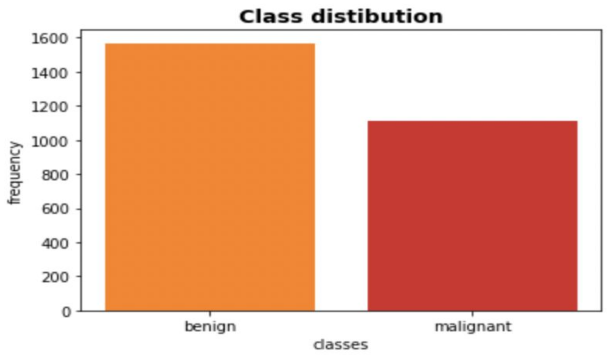
Abnormality classification class distribution



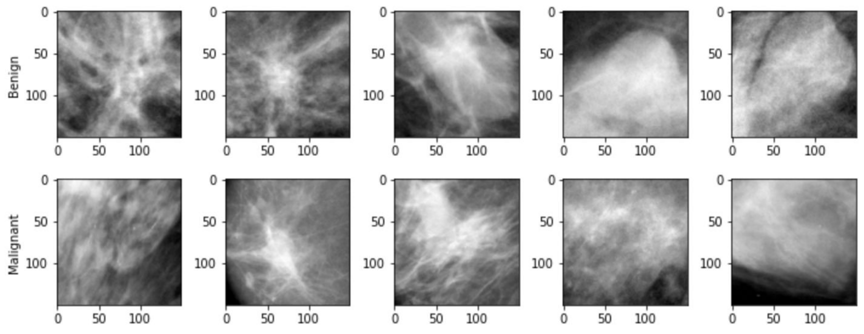
Abnormality classification sample images



Abnormality diagnosis class distribution



Abnormality diagnosis sample images



We performed almost the same pre-processing to the images for each project task. First, we normalized the images between 0 and 1 by multiplying the tensors for 1/65535. Second, we added the channel dimension to the tensors. In the case of scratch CNNs, we used greyscale images adding the channel dimension, from 150x150 to 150x150x1. In the case of pre-trained CNNs instead, as these networks require RGB images as input, we replicated the tensor into three channels to have the same dimension of an RGB image, from 150x150 to 150x150x3, without explicitly convert it.

For the abnormality diagnosis task, we tried pre-processing techniques inspired by the literature to boost the performances of the classifiers. In particular, we used median filters to smooth the images and contrast enhancement. None of these techniques resulted in a boost in performances, so we decided to not use them.

The number of images in the dataset is too small to let train a CNN from scratch. Hence, we decided to use data augmentation during the trainings. Data augmentation is useful also to fight overfitting. From the literature, we saw that the most used image transformation are rotation, flipping and shearing. We decided to use transformations that not introduce many distortions in the images. In particular, we used random horizontal and vertical flip and random rotation of 90 degrees.

The training set was grouped by class (class 0 on top, class 1 at the end), so, to split it into training and validation, it was necessary to shuffle. The splitting ratio used is 0.8 for training and 0.2 for validation.
Since the binarized datasets are unbalanced, we used class weighting, weighting the performance measures (Precision, Recall, F1-score, AUC) accordingly. We used the AUC to compare the classifiers.

Scratch CNN

The first task of this project was to design, implement and train two CNNs from scratch.

Abnormality classification

We designed two CNNs, one with two convolutional layers and one with three convolutional layers. Both CNNs use MaxPooling operations and have 4 fully connected layers. The networks have the following structure:

- Convolution: Input shape = 150x150x1, kernel = 5x5, filters = 32, activation = ReLU;
- MaxPooling: pool size = 2x2;
- Convolution: kernel = 5x5, filters = 64, activation = ReLU;
- MaxPooling: pool size = 2x2;
- Convolution: kernel = 5x5, filters = 128, activation = ReLU; *
- MaxPooling: pool size = 2x2; *
- Dense: units = 512, activation = ReLU,
- Dense: units = 256, activation = ReLU,
- Dense: units = 128, activation = ReLU,
- Dense: units = 1, activation = Sigmoid.

* This layer is included only in the 3-layer-CNN.
The training is performed using early stopping on the validation accuracy (max value, patience 20 epochs) and max number of epochs 100. The training set is shuffled at each epoch.

We tested the two CNN to choose the best architecture. The result for the 2-layer-CNN is:

Batch size	**Learning rate**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**AUC**	**Optimizer**
32	1e-4	0.8363	0.8534	0.8363	0.8356	0.8424	RMSprop

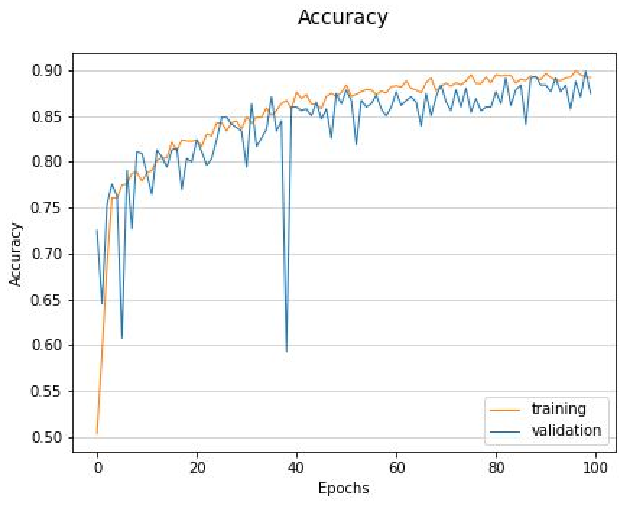
The result for the 3-layer-CNN is:

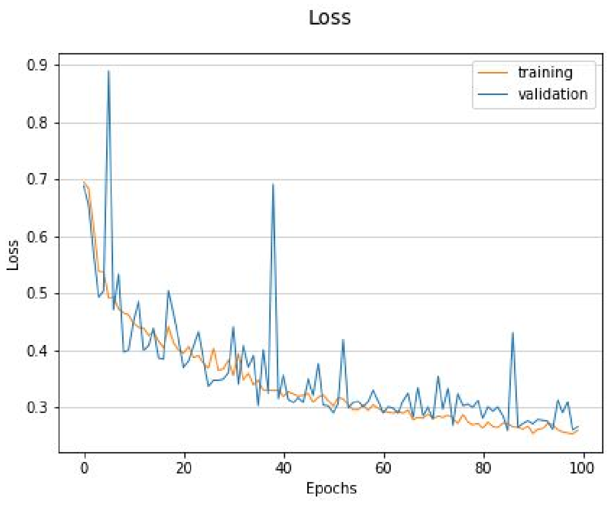
Batch size	Learning rate	Accuracy	Precision	Recall	F1-score	AUC	Optimizer
32	1e-4	0.8571	0.8603	0.8571	0.8562	0.8530	RMSprop

According to the AUC value, the best classifier is the 3-layer-CNN. To select a good set of hyperparameters (Batch size, Learning rate, Optimizer), we performed a manual search training the network with a different set of hyperparameters. The results are summarized in the following table:

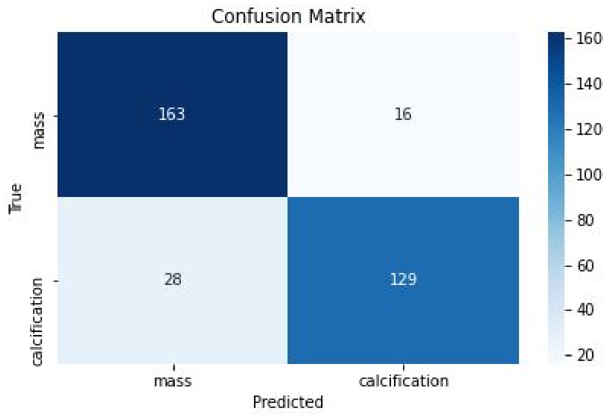
	Batch size	Learning rate	Accuracy	Precision	Recall	F1-score	AUC	Optimizer
1	16	1e-4	0.8571	0.8603	0.8571	0.8562	0.8530	RMSprop
2	**32**	**1e-4**	**0.8690**	**0.8703**	**0.8690**	**0.8685**	**0.8661**	**RMSprop**
3	64	1e-4	0.8452	0.8452	0.8452	0.8452	0.8445	RMSprop
4	32	1e-5	0.8065	0.8265	0.8065	0.8054	0.8133	RMSprop
5	32	1e-4	0.8303	0.8407	0.8303	0.8301	0.8349	Adam

According to these results, the best set of hyperparameters for this network is set 2, that gives us an AUC value of ~86%. For brevity, we report only the plots of the training of this model:





The confusion matrix of the classifier is:



Abnormality diagnosis

The network used for this task is the 3-layer-CNN used for the previous task. We trained the network in the same way as before but with different pre-processing on the images. The first train is performed only with normalization. The second train is performed using normalization, a median smoothing filter and contrast adjustment.

Training with normalization:

Batch size	**Learning rate**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**AUC**	**Optimizer**
16	1e-4	0.6398	0.6836	0.6398	0.6483	0.6520	RMSprop

Training with smoothing and contrast adjustment:

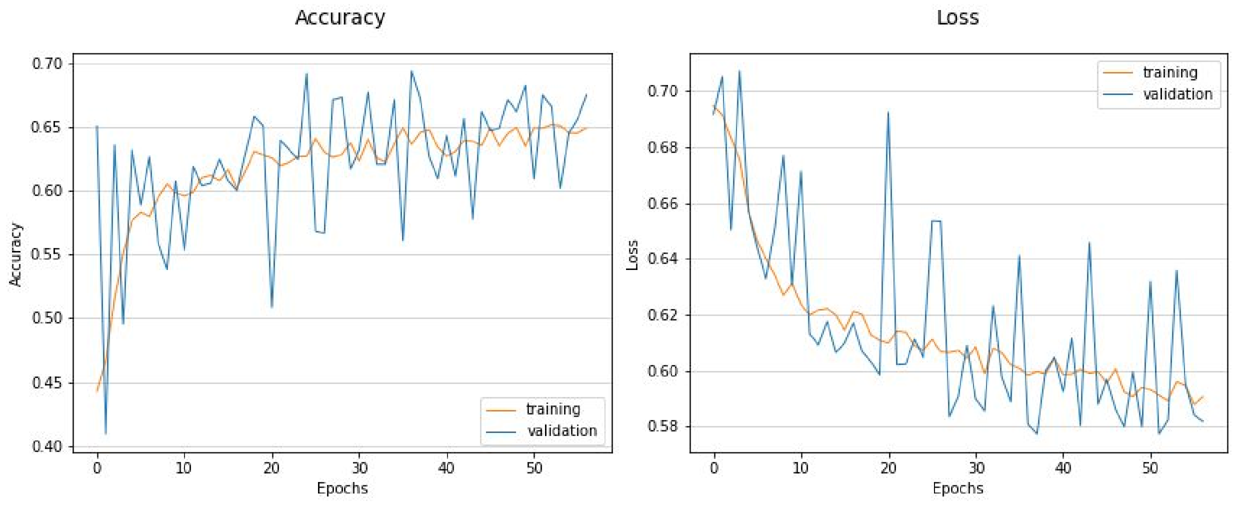
Batch size	**Learning rate**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**AUC**	**Optimizer**
16	1e-4	0.6488	0.6624	0.6488	0.6538	0.6310	RMSprop

According to these results, we decided to remove smoothing and contrast adjustment.

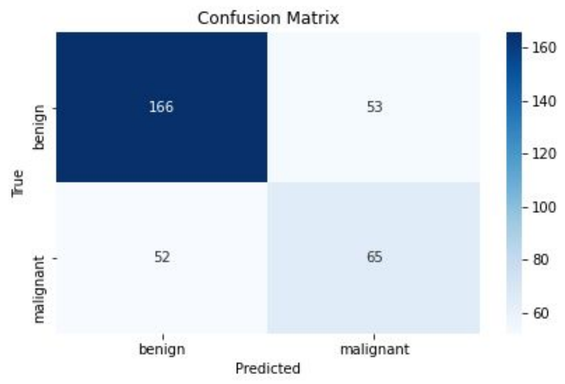
We then performed the hyperparameters search as before:

	Batch size	Learning rate	Accuracy	Precision	Recall	F1-score	AUC	Optimizer
1	16	1e-4	0.6666	0.6693	0.6666	0.6679	0.6368	RMSprop
2	32	1e-4	0.6547	0.6666	0.6547	0.6592	0.6356	RMSprop
3	**64**	**1e-4**	**0.6875**	**0.6881**	**0.6875**	**0.6878**	**0.6567**	**RMSprop**
4	64	1e-5	0.6369	0.6272	0.6369	0.6309	0.5861	RMSprop
5	64	1e-4	0.6428	0.6442	0.6428	0.6435	0.6085	Adam

The best set of hyperparameters in this case is 3. The result of the training are:



We can notice a fluctuation in the validation accuracy and loss curves. The reason is that the validation set could be not representative of the training set (the validation set could be too small for this task) because we used a 20% of the data for the validation set. However, we noticed that the situation is the same using a split of 30%. For this task, the dataset seems to be too small, so there are no solutions to this problem. The confusion matrix is:



We can notice from the confusion matrix that the classifier is not good at recognizing malignant abnormalities.

Pre-trained CNN

The second task of the project was to use pre-trained CNNs to perform the classification tasks. We have two options for transfer learning: use the pre-trained network as feature extractor (deep features) or fine-tune the pre-trained network. The size of the dataset and the similarity to the one used to train the CNNs matters in the choice of the strategy to follow. We have two choices to make, one is about the pre-trained network to use and one about the strategy.
We decided to use two different pre-trained CNNs: VGG16 and InceptionV3. We decided to use these networks because they are two of the most used networks in the literature. The first has a structure very similar to our scratch model, the second has a different structure and, according to the literature, has good results, so it worth a try. About the strategy, as our dataset is small and very different from ImageNet (on which VGG16 and InceptionV3 are trained), we decided to use them as feature extractors. Moreover, for the same reason, we decided to use only the first 3 layers of the pre-trained networks to exploit the most generic filters.

Abnormality classification

We used both VGG16 and InceptionV3 to train MLP classifiers with the same architecture. The architecture of the MLPs is:

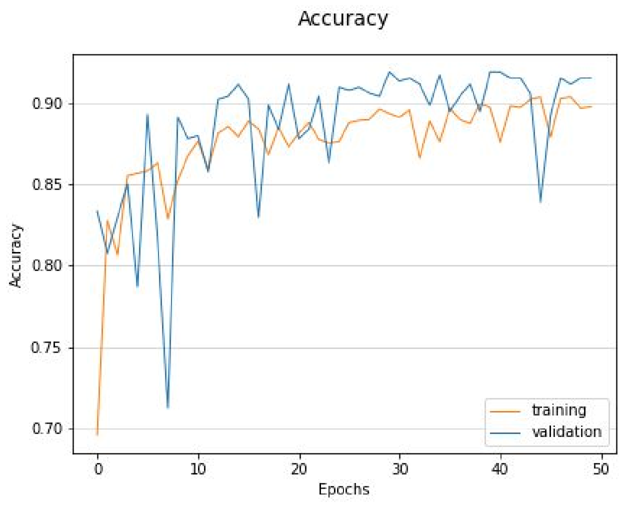
- Dense: units = 512, activation = ReLU,
- Dense: units = 256, activation = ReLU,
- Dense: units = 128, activation = ReLU,
- Dense: units = 1, activation = Sigmoid.

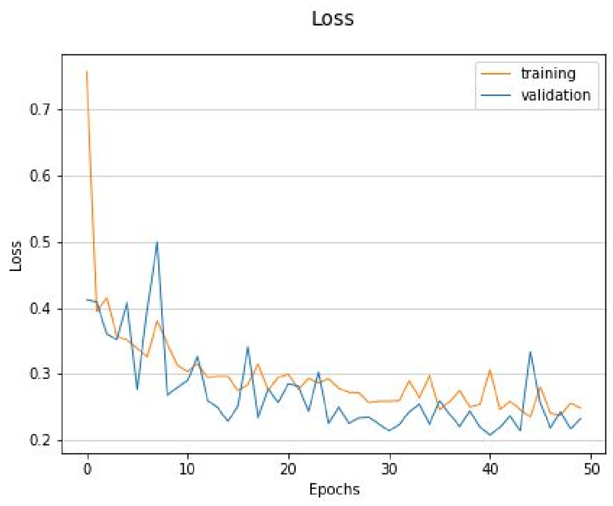
The training is performed with early stopping as in the scratch task. We performed hyperparameters search for both networks.

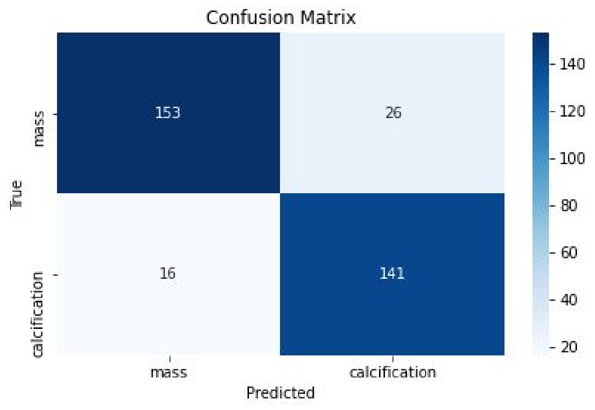
Training with VGG16:

	Batch size	**Learning rate**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**AUC**	**Optimizer**
1	16	1e-4	0.8690	0.8692	0.8690	0.8688	0.8673	Adam
2	32	1e-4	0.8422	0.8560	0.8422	0.8418	0.8476	Adam
3	64	1e-4	0.8690	0.8694	0.8690	0.8691	0.8692	Adam
4	**64**	**1e-5**	**0.875**	**0.8768**	**0.875**	**0.8751**	**0.8764**	**Adam**
5	64	1e-5	0.8601	0.8616	0.8601	0.8602	0.8612	RMSprop

The best set of hyperparameters for this model is set 4. As before, we report the plots of this training and the confusion matrix:



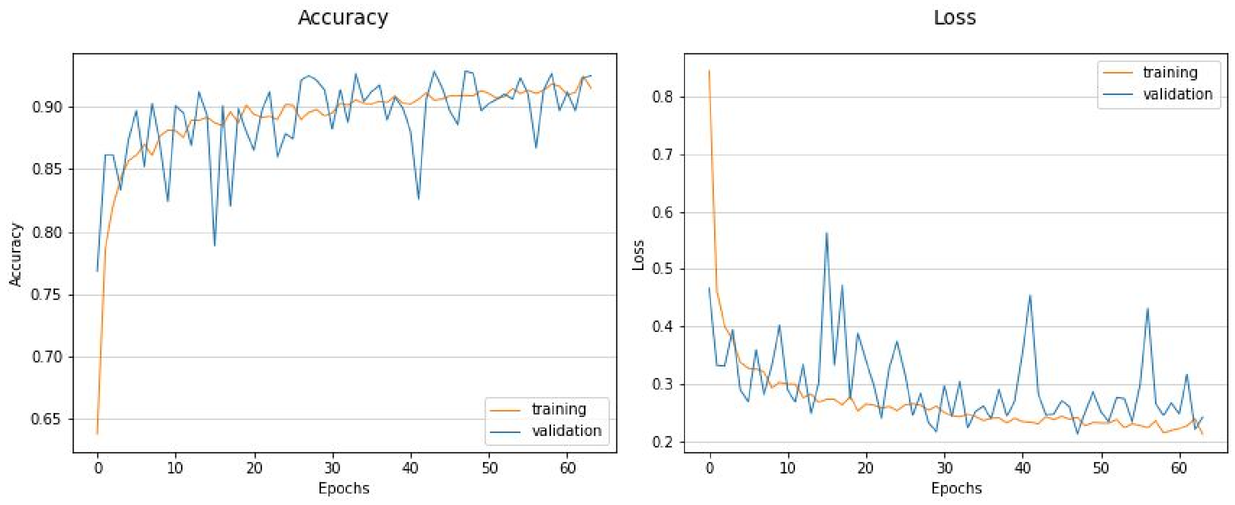


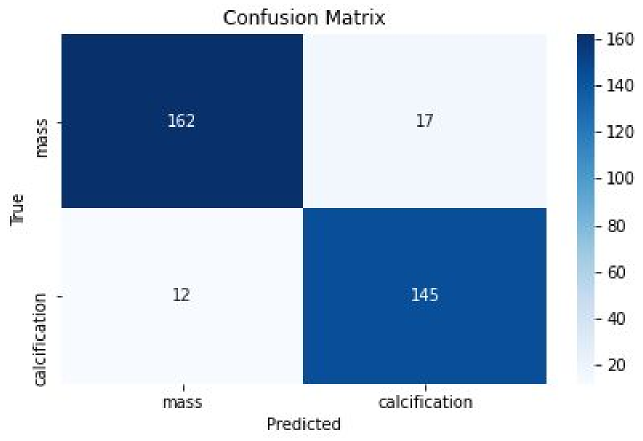


Training with InceptionV3:

	Batch size	Learning rate	Accuracy	Precision	Recall	F1-score	AUC	Optimizer
1	16	1e-4	0.9017	0.9017	0.9017	0.9017	0.9007	Adam
2	32	1e-4	0.8988	0.9000	0.8988	0.8989	0.8999	Adam
3	64	1e-4	0.8988	0.8995	0.8988	0.8988	0.8995	Adam
4	16	1e-5	0.8928	0.8946	0.8928	0.8929	0.8943	Adam
5	**16**	**1e-4**	**0.9136**	**0.9142**	**0.9136**	**0.9137**	**0.9142**	**RMSprop**

The best set of hyperparameters is set 5. Training plots and confusion matrix:





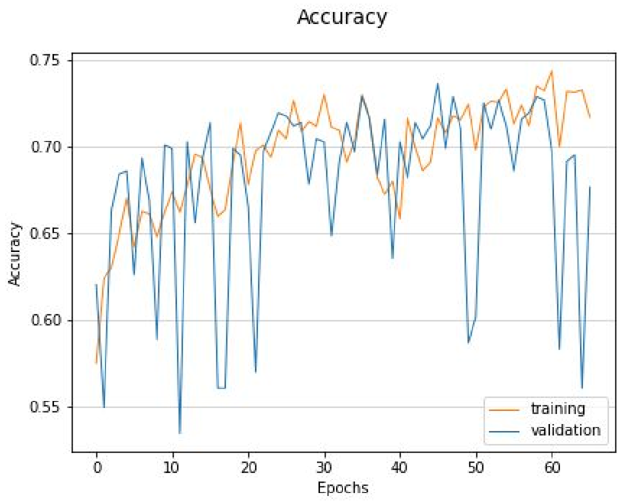
We can highlight that the classifier with InceptionV3 achieves slightly better results in contrast to the one with VGG16.

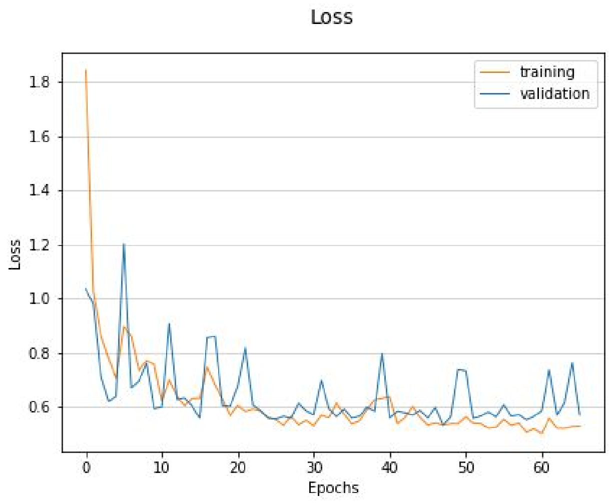
Abnormality diagnosis

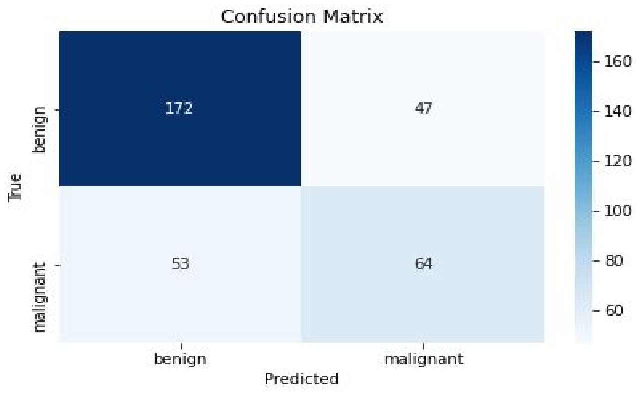
We used the same MLP also for this task, trained in the same way. Training with VGG16:

	Batch size	Learning rate	Accuracy	Precision	Recall	F1-score	AUC	Optimizer
1	16	1e-4	0.6726	0.6825	0.6726	0.6764	0.6533	Adam
2	**32**	**1e-4**	**0.7023**	**0.6990**	**0.7023**	**0.7004**	**0.6661**	**Adam**
3	64	1e-4	0.6964	0.6976	0.6964	0.6970	0.6676	Adam
4	32	1e-5	0.7083	0.6990	0.7083	0.7010	0.6588	Adam
5	32	1e-4	0.6726	0.6592	0.6726	0.6624	0.6155	RMSprop

We selected the set 2 instead of the 3 even if the AUC is slightly lower because all the other measures are higher. Training plots and confusion matrix:



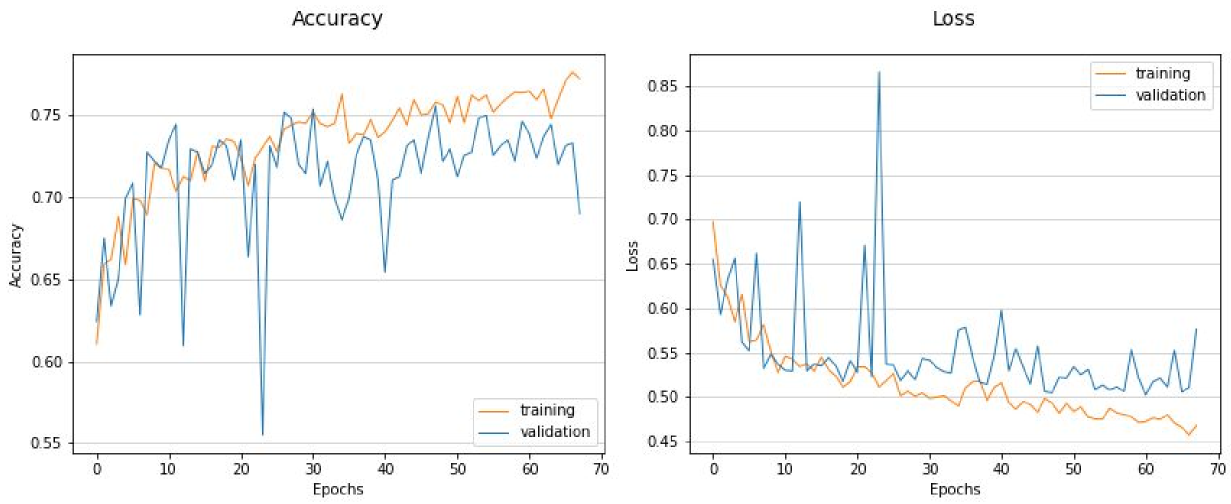


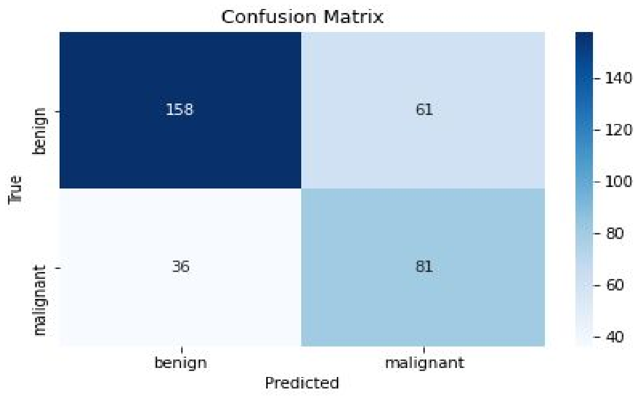


Training with InceptionV3:

	Batch size	Learning rate	Accuracy	Precision	Recall	F1-score	AUC	Optimizer
1	**16**	**1e-4**	**0.7113**	**0.7294**	**0.7113**	**0.7165**	**0.7068**	**Adam**
2	32	1e-4	0.6726	0.7020	0.6726	0.6797	0.6752	Adam
3	64	1e-4	0.7142	0.7179	0.7142	0.7158	0.6912	Adam
4	16	1e-5	0.6755	0.6911	0.6755	0.6807	0.6635	Adam
5	16	1e-4	0.7232	0.7273	0.7232	0.7249	0.7020	RMSprop

Here the difference is higher in the AUC, so we selected the set 1. Training plots and confusion matrix:





The classifier with InceptionV3 performs better than the one with VGG16 also in this case.

Baseline CNN

The third task of the project was to implement a solution for one of the two classification task exploiting the baseline patches. As suggested, we used the siamese network for the abnormality classification task. The siamese network is composed of two twins CNN that share the weights. These networks extract the features from two images given as input. The features are then merged into a distance layer. The architecture of the twins is:

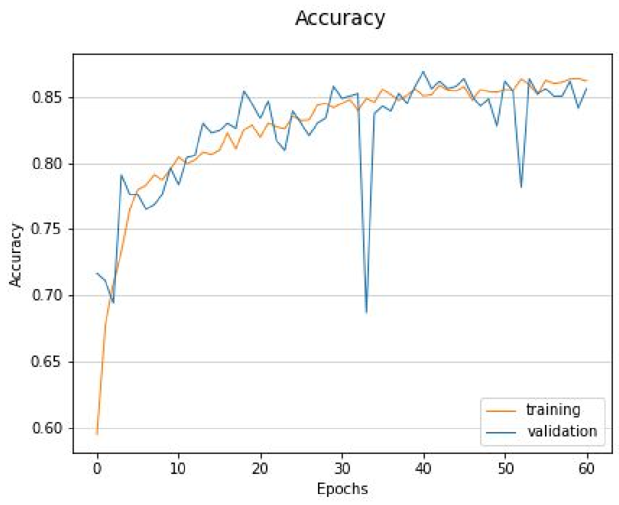
- Convolution: Input shape = 150x150x1, kernel = 5x5, filters = 32, activation = ReLU;
- MaxPooling: pool size = 2x2;
- Convolution: kernel = 5x5, filters = 64, activation = ReLU;
- MaxPooling: pool size = 2x2;
- Convolution: kernel = 5x5, filters = 128, activation = ReLU;
- MaxPooling: pool size = 2x2;
- Dense: units = 512, activation = ReLU.

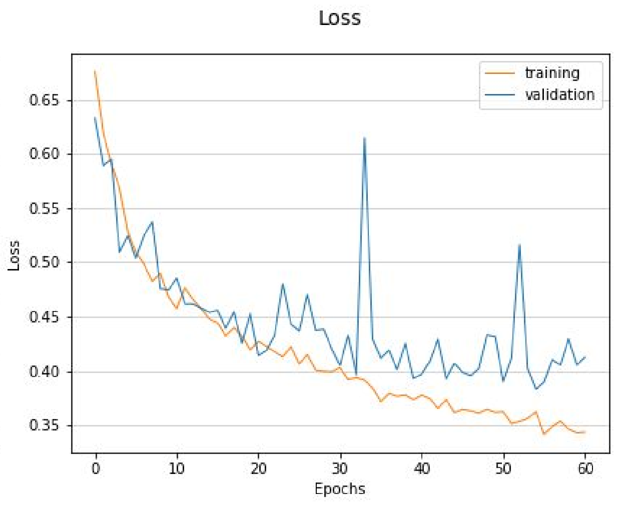
In the distance layer, we just made the absolute value of the difference of the features.
The input dataset of this network is slightly modified. We took the baseline patches and the respective abnormality patches, we merged them into an array assigning as class label the type of abnormality. In this way, one of the twins will receive the baseline patch and the other one the corresponding abnormality patch. As in the other tasks, we used early stopping.

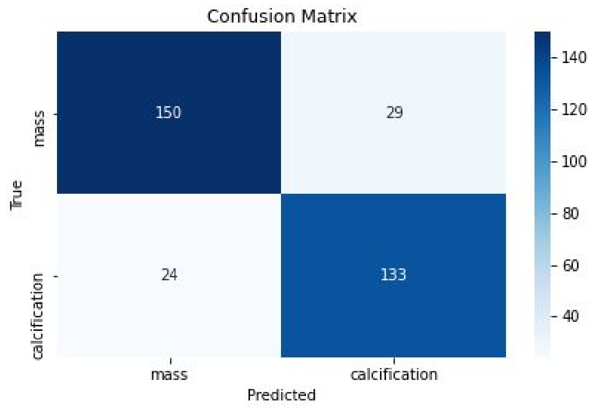
The results of the training are:

	Batch size	Learning rate	Accuracy	Precision	Recall	F1-score	AUC	Optimizer
1	16	1e-4	0.8035	0.8072	0.8035	0.8018	0.7984	RMSprop
2	**32**	**1e-4**	**0.8422**	**0.8428**	**0.8422**	**0.8423**	**0.8425**	**RMSprop**
3	64	1e-4	0.8035	0.8036	0.8035	0.8031	0.8011	RMSprop
4	32	1e-5	0.7827	0.7834	0.7827	0.7829	0.7827	RMSprop
5	32	1e-4	0.8005	0.8009	0.8005	0.8006	0.8003	Adam

The best hyperparameters set is the second. Training plots and confusion matrix:







Ensemble

The last task of the project was to implement an ensemble method, and we used the best classifiers from each task of this project to do so. Furthermore, we decided to not use the siamese network in this task because it has a lower performance compared with the other networks.
We have three possible choices for the ensemble method: bagging, boosting and stacking. Since the performances of the classifiers are very different, in most cases, we decided to do boosting giving each classifier a weight according to its AUC score. We used the AUC score instead of the accuracy because we are in the case of an unbalanced dataset.

Performance in abnormality classification:

Accuracy	**Precision**	**Recall**	**F1-score**	**AUC**
0.9017	0.9017	0.9017	0.9017	0.9011

Performance in abnormality diagnosis:

Accuracy	**Precision**	**Recall**	**F1-score**	**AUC**
0.6845	0.7081	0.6845	0.6031	0.5569

Conclusions

For this project, we performed two classification tasks related to breast cancer: abnormality classification and abnormality diagnosis. The tasks are performed using different techniques: training from scratch, transfer learning, siamese and ensemble.
As expected, the classifiers that exploit the pre-trained networks outperform the classifiers trained from scratch. This is due to different factors: the network architecture, the dimension of the dataset used for training and the training method. The siamese network and the ensemble classifier achieve good performances but still lower than the pre-trained networks. The best classifier for both classification tasks is the one that exploits the InceptionV3, achieving results similar to the results found in the literature.